Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Plevel

Predelava Magento sistema za upravljanje vsebin spletne trgovine v sistem za evidenco dela zaposlenih

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Aleš Smrdel

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Naslov: Predelava Magento sistema za upravljanje vsebin spletne trgovine v sistem za evidenco dela zaposlenih

Tematika naloge:

V diplomski nalogi razvijte sistem za evidenco dela zaposlenih. Sistem naj bo implementiran kot spletna aplikacija. Razviti sistem naj implementira nekatere najbolj uporabljane funkcionalnosti, ki jih ponujajo drugi tovrstni sistemi. Pri implementaciji sistema pa kot osnovo vzemite sistem za upravljanje vsebin spletne trgovine Magento in ga prilagodite zahtevam evidence dela. Ker ta sistem v osnovi ni namenjen vodenju evidence dela, ga najprej reducirajte in mu odstranite funkcionalnosti, ki se jih ne potrebuje. Tako dobljen reduciran sistem pa nadgradite z moduli, ki bodo implementirali potrebne funkcionalnosti. Pri implementaciji sistema se omejite na tehnologije na strani strežnika in odjemalca, ki jih že uporablja sistem Magento. Predstavite tudi prikaz razvoja modulov, s katerimi dodajamo funkcionalnosti v sistem. Pri razvoju izgleda spletne aplikacije pa bodite pozorni na to, da se uporabniški vmesnik aplikacije primerno prikaže na različnih napravah in dimenzijah zaslonov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Rok Plevel sem avtor diplomskega dela z naslovom:

Predelava Magento sistema za upravljanje vsebin spletne trgovine v sistem za evidenco dela zaposlenih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Aleša Smrdela,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Cerkljah na Gorenjskem, dne 11. septembra 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Alešu Smrdelu za pomoč, svetovanje in spodbudo pri izdelavi, še posebej pisanju, diplomskega dela. Zahvaljujem se tudi sodelavcem in direktorju podjetja IB-CADDY d.o.o., kjer sem v času izdelave opravljal študensko delo, še posebej pa sodelavcu Mihu, ki je pripomogel k idejni zasnovi diplomskega dela. Vsem svojim bližnjim se zahvaljujem za podporo in spodbudo v času študija.

Vsem svojim.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CMS	Content Management System	Sistem za upravljanje vsebin
MVC	Model-View-Controller	Model-pogled-krmilnik
HTML	Hyper Text Markup Language	Jezik za označevanje hipertekstovni
PHP	PHP: Hypertext Preprocessor	PHP: hipertekstovni predprocesor
XML	Extensible Markup Language	Razširljiv označevalni jezik
ORM	Object-relational mapping	Objekto-relacijska preslikava
SQL	Structured Query Language	Strukturirani povpraševalni jezik
EAV	Entity-Attribute-Value	Entiteta-atribut-vrednost
CRUD	Create, Read, Update, Delete	Kreiranje, branje, posodobitev, brisanje
CSS	Cascading Style Sheets	Kaskadne stilske podloge
DOM	Document Object Model	Model objekta dokumenta
AJAX	Asynchronous JavaScript and XML	Asinhroni JavaScript in XML
HTTP	Hypertext Transfer Protocol	Protokol za prenos hiperteksta
RDBMS	Relational Database Management System	Sistem za upravljanje z relacijsko podatkovno bazo
RSS	Rich Site Summary	Zgoščeni povzetek strani

Povzetek

Diplomska naloga obsega predelavo Magento sistema za upravljanje vsebin spletne trgovine v sistem za evidenco dela zaposlenih.

V prvem delu diplomskega dela so najprej predstavljeni razlogi za izbor Magento sistema, prikazane pa so tudi nekatere že obstoječe spletne rešitve za evidenco dela zaposlenih. Sledi predstavitev Magento sistema, oziroma princip delovanja njegove MVC arhitekture. Nato sledi opis orodij, s katerimi je izdelan Magento sistem, ter opis orodij, ki smo jih dodali za učinkovitejše in uporabniku prijaznejše delovanje našega sistema za evidenco dela.

V drugem delu je opisan postopek izdelave, ki se začne s postopkom redukcije Magento sistema, v bolj splošen sistem za razvoj spletnih aplikacij. S pomočjo primera implementacije glavnega modula, ki služi sistemu za evidenco dela zaposlenih, prikažemo razvoj modulov v Magento sistemu in predstavimo tudi primer razširitve obstoječe funkcionalnosti Magento sistema. Sledi še opis vseh ostalih funckionalnosti, ki smo jih implementirali za sistem za evidenco dela, ter opis izdelave dizajna, ki poskuša slediti najnovejšim trendom pri razvoju spletnih aplikacij.

V zadnjem delu pa so predstavljeni cilji nadaljnega razvoja aplikacije, kjer nekateri temeljijo na že obstoječih funkcionalnostih Magento sistema.

Ključne besede: Magento sistem za upravljanje vsebin spletne trgovine, sistem za evidenco dela zaposlenih, MVC arhitektura, modul, XML, PHP.

Abstract

This thesis comprises a rework of the Magento e-commerce content management system into an employee attendance tracking system.

In the first part of the thesis the reasons for selecting Magento system are presented and also some already existing web solutions for employee attendance tracking are shown. Next is the presentation of the Magento system, that is, principles of its MVC architecture. After that follows the description of the tools on which Magento is based upon and some of the tools we used to make our employee attendance tracking software better.

The second part describes the development process, which starts with the reduction of the Magento system, to a more general system for web based application development. With the help of the implementation process of our main module, which serves the purpose of the employee attendance tracking system, we describe the module development process in Magento system and present an example of extending a core functionality. Next, the presentation of the remaining functionalities implemented for our employee attendance tracking system follows, and the description of the design, which attempts to follow the newest trends in web application development.

In the last part we present the goals of the future development of the application, where some of them are based on already existing functionalities of the Magento system.

Keywords: Magento e-commerce content management system, employee attendance tracking system, MVC architecture, module, XML, PHP.

Poglavje 1

Uvod

Razvoj spletnih tehnologij je vedno hitrejši in razlogov za to je veliko. Med najpomembnejše gotovo spada vse lažji dostop do svetovnega spleta ter vse večja uporaba mobilnih naprav, kot so tablice in pametni telefoni [1]. Razvijalci spletnih aplikacij, torej aplikacij, do katerih dostopamo preko spleta in se na strani odjemalca izvajajo v brskalniku, se morajo osredotočiti le na to, da se aplikacija pravilno izvaja v večini najbolj uporabljenih brskalnikov. V primerjavi z navadnimi aplikacijami so tako poenostavljeni razvoj in posodobitve. Aplikacije, ki domujejo na spletu, imajo torej številne prednosti in vse več aplikacij se seli v spletne brskalnike.

Vsak delodajalec je v skladu z Zakonom o evidencah na področju dela in socialne varnosti [2] dolžan skrbeti za in voditi številne podatke, povezane z zaposlenimi. Pri velikem številu podjetji ni namen vodenja le izognitev kazni ob neupoštevanju zakona, ampak lahko vodenje take evidence tudi olajša poslovanje, saj od delodajalcev zahteva red in organiziranost ter jim prihrani veliko časa pri izpolnjevanju obveznosti do zaposlenih.

Avtor diplomskega dela sem v času opravljanja prakse sodeloval pri podjetju, ki kot mnoga druga v ta namen uporablja spletno aplikacijo. Sistem je v uporabi že več kot pet let, tehnologija pa je v tem času močno napredovala, tako da bi lahko bil deležen prenove.

Med opravljanjem prakse sem imel priložnost izdelati spletno trgovino z

zelo znanim Magento sistemom za upravljanje vsebin (CMS - Content Management System) za elektronsko poslovanje [3] in zanj dodelati tudi nekaj modulov. Pojavila se je ideja, da bi sistem predelali v bolj splošen spletni sistem, s katerim bi bilo mogoče na novo izdelati sistem za evidenco dela zaposlenih, reducirano verzijo pa bi kasneje lahko uporabili tudi za drugačne produkte.

V naslednjem poglavju bomo najprej predstavili razloge, ki so nas vodili v izbiro Magento sistema kot osnovo sistema za evidenco dela, kjer upoštevamo tudi trenuten sistem za evidenco dela, ki ga uporablja podjetje. Prav tako bomo prikazali posamezne obstoječe spletne rešitve, iz katerih smo prevzeli nekatere ideje. V tretjem poglavju pride na vrsto podroben opis Magento sistema, kjer se bomo, kolikor se da, izognili opisu funkcionalnosti namenjenih spletni trgovini. V naslednjem poglavju bomo spoznali programska orodja in tehnologije, ki jih uporablja Magento sistem, ter orodja, ki smo jih dodali med razvojem. Peto poglavje je namenjeno izdelavi aplikacije, kjer bomo opisali tudi funkcionalnosti. Na začetku petega poglavja opisujemo postopek reduciranja sistema Magento v splošen sistem za izdelavo spletnih aplikacij, ki bo namenjen uporabnikom z dobrim znanjem spletnega programiranja. Nato bomo na primeru implementacije enega izmed najpomembnejših modulov, ki je namenjen dnevnemu vnosu aktivnosti zaposlenih, podrobno opisali njegovo izdelavo in poskušali skupaj z vsebino tretjega poglavja jasno prikazati postopek izdelave modulov v Magento sistemu. V nadaljevanju petega poglavja prikažemo še ostale funkcionalnosti, kjer se osredotočamo na bolj zanimive dele. Za konec petega poglavja bomo opisali še dizajn, kjer poskušamo slediti najnovejšim trendom in trenutno zelo pomembnemu odzivnemu dizajnu (angl. responsive design) za mobilne naprave. Zadnje poglavje je namenjeno zaključkom in funkcionalnostim, ki nam jih ni uspelo dokončati oziroma jih še želimo implementirati.

Poglavje 2

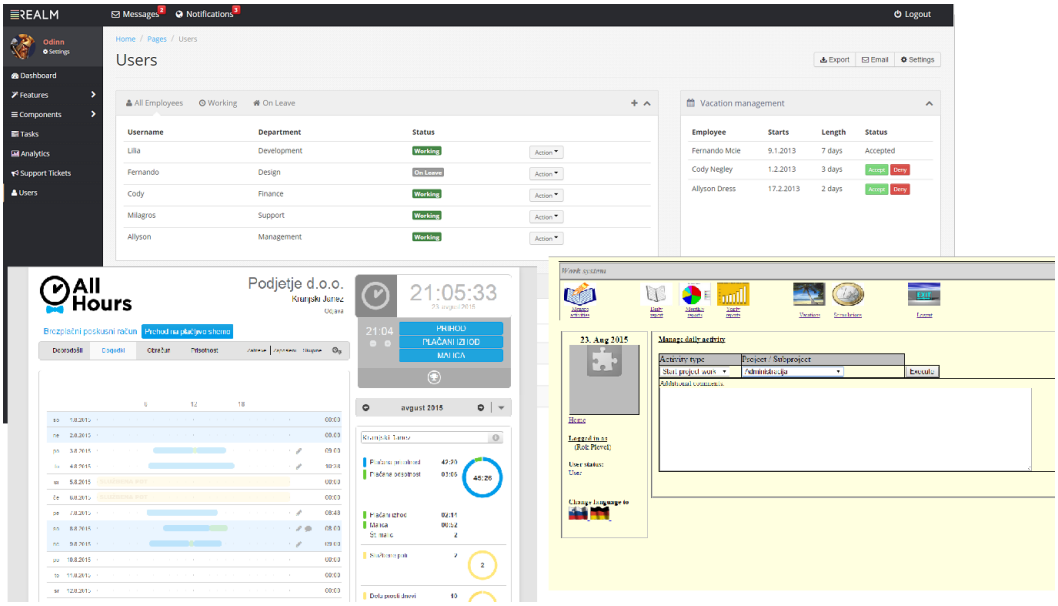
Razlogi za izbor Magento sistema in pregled obstoječih rešitev

Glavni cilj diplomskega dela je izdelati sistem za evidenco dela zaposlenih. Pretekle izkušnje razvijanja modulov za Magento sistem so nas privedle do ideje, da bi lahko za izdelavo takšnega sistema uporabili kar Magento sistem. Njegove glavne prednosti so fleksibilnost, možnost nadgrajevanja in hitrost delovanja. Naštete prednosti so predvsem posledica dobro zastavljene arhitekture tipa model-pogled-krmilnik (MVC - Model-View-Controller), ki bo podrobneje opisana v naslednjem poglavju. Zanimiva je tudi njegova datotečna struktura, ki ločuje jedro kode od tiste, ki jo kasneje dodajajo razvijalci. Zaradi te strukture se je kasneje pojavila še ideja, da bi cilj predelave stremel tudi k bolj splošnemu sistemu za izdelavo spletnih aplikacij. V ta namen je potrebno Magento sistem najprej zreducirati na neko osnovo in kodo sistema za evidenco dela ločiti od te osnove, pri tem pa moramo paziti, da ne odstranimo obstoječih funkcij, ki bi nam morebiti lahko služile za naš sistem, saj niso vse zgolj za namen spletne trgovine.

Najprej moramo dobro zasnovati, kako bo naš sistem deloval in kaj nameravamo implementirati. Za zgled smo vzeli star sistem podjetja, ki je izdelan

še kot velika HTML (Hyper Text Markup Language) tabela. Glavne funkcionalnosti tega sistema je bilo potrebno obdržati, saj je podjetje navajeno takšnega delovanja. Med zahtevane funcionalnosti sodijo prijava in odjava na delo, kjer izberemo tudi dejavnost, ki jo ta dan opravljamo, odhod na odmor, in mesečna ter letna evidenca opravljenih ur. Star sistem je sam po sebi precej skop, kar pa nam daje ogromno možnosti nadgradnje.

Pred začetkom samega načrtovanja dela smo si ogledali že obstoječe rešitve na spletu. Takšnih sistemov je kar veliko in nudijo ogromno funkcionalnosti, ki jih za naš sistem niti ne potrebujemo. Zato pa smo si lahko izmed množice različnih funkcionalnosti, ki jih nudijo tovrstni sistemi, izbrali tiste, ki so za nas najbolj pomembne. Na sliki 2.1 so prikazani trije sistemi, po katerih smo se najbolj orientirali. Funkcionalnost je povzeta po starem sistemu (na sliki 2.1 desno spodaj) ter po sistemu AllHours [4] (slika 2.1 levo spodaj), navdih za izgled pa smo našli v sistemu Realm [5] (na sliki 2.1, zgoraj).



Slika 2.1: Nekateri obstoječi sistemi za evidenco dela zaposlenih.

Poglavje 3

Magento - odprtokodni sistem za upravljanje vsebin

3.1 Kaj je Magento sistem

Magento sistem [6] je platforma za izdelavo spletne trgovine (angl. e-commerce platform), ki temelji na odprtokodni tehnologiji, ki spletnim trgovcem ponuja fleksibilnost ter nadzor nad izgledom, vsebino in funkcionalnostjo njihove spletne trgovine. Na kratko povedano, Magento sistem trgovcem omogoča možnost izdelave spletne trgovine, prirejene njihovim poslovnim potrebam. Intuitiven administrativni vmesnik omogoča dobro trženje, orodja za nadzor prodajanih izdelkov, sledenje prodaji in druge bolj ali manj uporabljene funkcionalnosti. Ravno administrativni vmesnik je tisto, kar bomo priredili našim potrebam, vse sledi spletne trgovine pa bomo odstranili.

Magento sistem je sistem za upravljanje vsebin, izdelan v programskem jeziku PHP. Temelji na MVC arhitekturi in je razdeljen na skupine modulov. Vsak modul ločuje področja funkcionalnosti in se poskuša izogibati odvisnosti od drugih modulov, kar prinaša ogromno fleksibilnost in s tem tudi mero kompleksnosti. Sveža namestitev trenutno sestavlja okoli 30.000 datotek in več kot 1.2 milijona vrstic kode.

3.2 Zend Framework - osnova Magento sistema

Zend Framework [7] je odprtokodna rešitev za razvoj spletnih aplikacij z uporabo programskega jezika PHP verzije 5.3 in višje. Uporablja popolnoma objektno usmerjeno kodo in večino novejših značilnosti jezika PHP, kot so imenski prostor, poznejše statične vezave, lambda funkcije in procedure. Struktura komponent sledi SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion) objektno usmerjenem principu načrtovanja. Akronim SOLID [8] opisuje pet principov objektno usmerjenega programiranja, kjer želimo ustvariti sistem, ki ga lahko skozi daljše obdobje preprosto vzdržujemo in razširimo. Vsaka komponenta mora biti čim manj odvisna od ostalih. Takšna ohlapno povezana arhitektura omogoča razvijalcem uporabljati katerokoli komponento, kar so ustvarjalci sistema poimenovali kot "uporabi-po-meri" načrtovanje.

Sistem implementira splošno znan pristop načrtovanja MVC. Podjetje Varien, kasneje Magento Inc., se je za razvoj Magento sistema, kjer ogrodje **Zend** služi kot osnova, odločilo na podlagi naslednjih komponent:

- *Zend_Cache*,
- *Zend_Acl*,
- *Zend_Locale*,
- *Zend_DB*,
- *Zend_Soap*,
- *Zend_Http*.
- *Zend_Pdf*,
- *Zend_Currency*,
- *Zend_Date*.

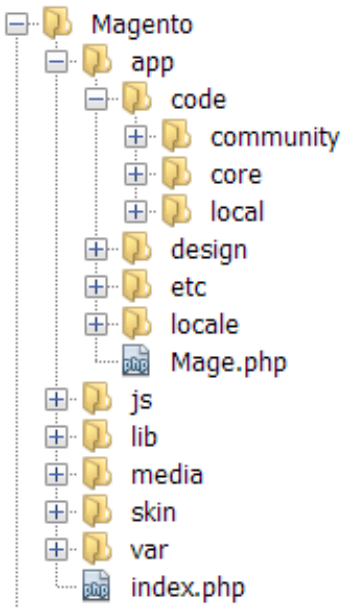
Skupno uporablja okoli petnajst **Zend** komponent. Knjižnici **Varien** ter kasneje **Mage** nekatere izmed njih direktno razširjata. Na primer, zgoraj omenjena *Zend_Cache* je za Magento sistem razširjena kot *Mage_Cache*. Pri razvoju in razširitvah so sledili naslednjim principom [9]:

- **Vzdrževalnost:** Kaže se s kodnimi območji, kjer je jedro kode ločeno od lokalnih sprememb ter zunanjih razširitev.
- **Nadgradljivost:** Modularnost omogoča implementacijo in nadgradnjo lokalnih razširitev ter razširitev tretjih oseb.
- **Fleksibilnost:** Preprosto prilagajanje in poenostavljen razvoj novih funkcionalnosti.

3.3 Struktura direktorijev

Magento struktura direktorijev je nekoliko drugačna od ostalih MVC aplikacij. Slika 3.1 prikazuje celotno arhitekturo direktorijev. Direktoriji, prikazani na sliki 3.1, so namenjeni:

- *app*: Jedro Magento kode, ki se še naprej deli na:
  - *code*: Tukaj najdemo celotno, večinoma PHP, kodo aplikacije, ki se deli na sklope *community* (za module tretjih oseb), *core* in *local* (za lokalne spremembe).
  - *design*: Vsebuje datoteke predlog in XML (angl. Extensible Markup Language) datoteke za postavitev strani. Večinoma je to HTML koda.
  - *etc*: V tem direktoriju se nahajajo XML datoteke, ki Magento sistemu povedo, da modul sploh obstaja, v katerem kodnem območju se nahaja, morebitne odvisnosti od ostalih modulov in če je modul aktiven.



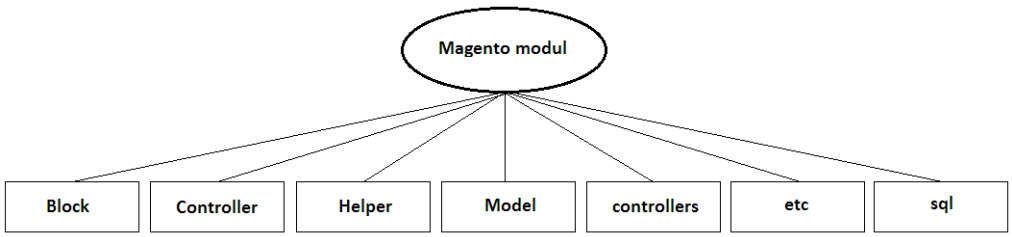
Slika 3.1: Magento glavna stuktura direktorijev.

  - *locale*: Služi CSV (Comma Separated Values) datotekam s prevodi.

- *js*: Vsebuje JavaScript knjižnice, ki jih uporablja Magento sistem.
- *media*: Direktorij je namenjen slikam produktov in drugim medijem, ki so dostopni na straneh aplikacije.
- *lib*: V tem direktoriju se nahajajo knjižnice tretjih oseb, kot sta **Zend** in **PEAR**. Vsebuje tudi knjižnice, ki so jih razvili pri Magento, in se nahajajo pod imenskim prostorom Varien in novejšim Mage.
- *skin*: Ta direktorij se naprej deli na ospredje (angl. frontend) in zaledje (angl. backend) ter ločuje različne teme, ki jih morebiti uporabljamo pri naši spletni aplikaciji. Večinoma je to CSS (Cascading Style Sheets) koda in tudi slike za izgled posamezne teme.
- *var*: Ta direktorij hrani začasne datoteke, kot so datoteke za predpomnjenje (angl. cache), datoteke sej (angl. session), razne datoteke za

uvoz in izvoz, itd.

Na sliki 3.1 seveda niso vidni vsi direktoriji. Magento je modularen sistem, kar pomeni, da sta aplikacija in njeno jedro deljena na več manjših modulov. Tipični direktorij za Magento modul je prikazan na sliki 3.2, kjer že lahko vidimo princip Magento MVC arhitekture, ki bo natančneje opisan v naslednjem podpoglavju.



Slika 3.2: Struktura direktorijev za posamezen Magento modul.

Na kratko preglejmo posamezni direktorij Magento modula:

- *Block*: Dodatna logična plast med krmilnikom in pogledom.
- *Controller*: Namenjeno je abstraktnim in razširjenim razredom krmilnikov v *controllers* direktoriju.
- *Helper*: Pomožni razredi, kjer lahko razširimo splošno funkcionalnost modula.
- *Model*: Krmilnik zahtevke obravnava, model pa služi za interakcijo z zbirko podatkov (podatkovno bazo).
- *controllers*: Definicija krmilnikov, kjer zajamemo različne spletne zahteve, večinoma od uporabnikov, lahko pa tudi od drugih funkcij sistema.
- *etc*: Konfiguracija modulov v obliki XML datotek.
- *sql*: Direktorij vsebuje namestitvene datoteke in datoteke za nadgradnjo strukture modula v podatkovni bazi.

3.4 Magento MVC arhitektura

Najbolj znana MVC arhitektura je MVC arhitektura temelječa na konvenciji. Kadar želimo implementirati na primer nov model ali krmilnik, je datoteko oziroma razred zanj potrebno dodati na ustrezno mesto, ki je določeno glede na konvencijo. MVC sistem jo nato samodejno zazna.

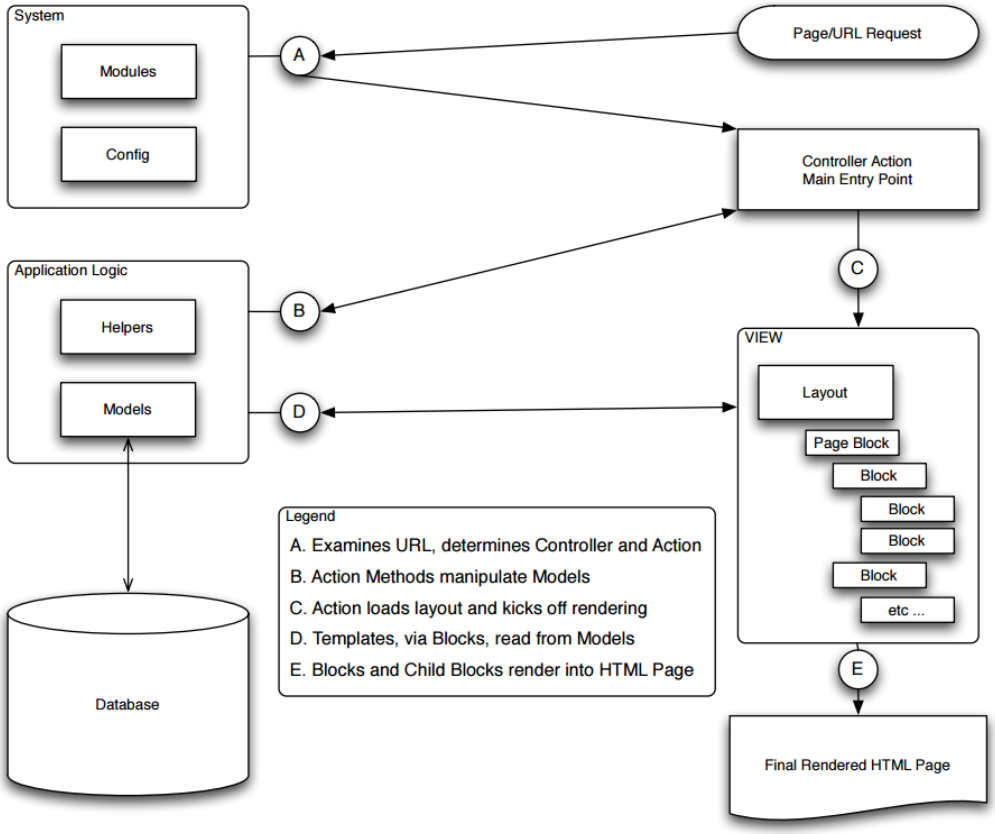
Magento uporablja MVC arhitekturo temelječo na konfiguraciji [10]. Pri takšni arhitekturi samo kreiranje datotek v jedro kode ni dovolj. Sistemu je potrebno eksplicitno povedati o novem razredu ali skupini razredov. V ta namen Magento sistem uporablja konfiguracijske datoteke *config.xml*, znotraj direktorija *etc*. Konfiguracijska datoteka za posamezen modul vsebuje vse glavne informacije, ki jih ta modul potrebuje za delovanje. Čeprav se to zdi zgolj kot dodatno delo, nam prinaša veliko večjo fleksibilnost in zmogljivost posameznih modulov. Tukaj lahko omenimo tudi *system.xml* datoteko, ki služi specifikaciji konfiguracije modula v zaledju aplikacije.

V naslednjih podpoglavjih bomo natančneje opisali delovanje Magento MVC arhitekture, ki je tudi prikazana na sliki 3.3. Vsebino zgoraj omenjenih konfiguracijskih datotek ter tudi vsebino datotek za modele, za krmilnike in za pogled aplikacije bomo obravnavali v petem poglavju na primerih naše aplikacije.

3.4.1 Model

Za dostop do podatkov v podatkovni bazi, uporablja Magento sistem ORM (Object-Relational Mapping) pristop. *Zend_Db* nam sicer ponuja neposreden dostop do baze z uporabo SQL poizvedb, vendar po večini v ta namen uporabljamo modele, kjer je za dostop do baze dovolj zgolj PHP koda. Poznamo dva tipa modelov [11]:

- **Preprosti modeli**: Implementacija modela definira povezavo enega objekta z eno tabelo v podatkovni bazi. Atributi objekta se natančno ujemajo z vsakim poljem v strukturi tabele.



Slika 3.3: Elementi in potek delovanja Magento MVC arhitekture [12].

- **EAV (Entity-Attribute-Value) modeli**: Takšen tip modela služi entitetam z dinamičnim številom atributov, ki se lahko nahajajo v različnih tabelah znotraj podatkovne baze.

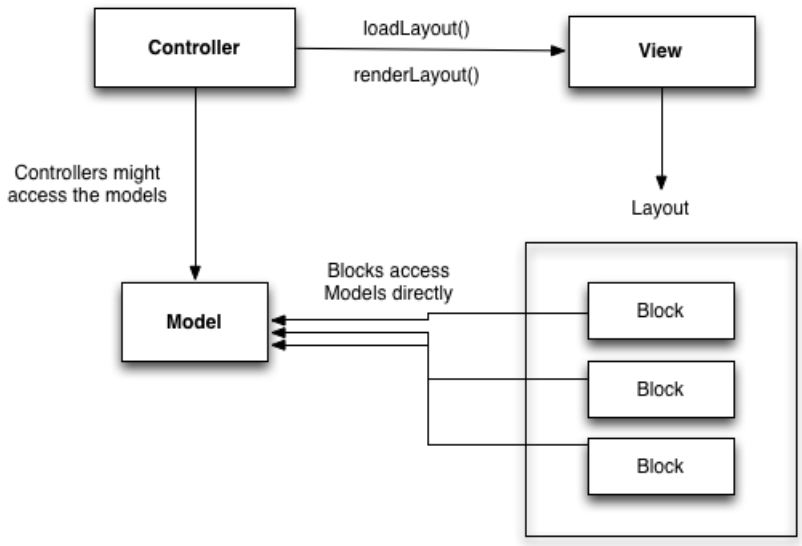
Model je razdeljen na dva dela in sicer na poslovno logiko ter na sredstva za interakcijo s podatkovno bazo. Magento model ne vsebuje nobene kode, kjer bi dostopali do podatkov v bazi. V ta namen uporablja razred *modelResource*. Tako nam Magento sistem teoretično omogoča uporabo druge podatkovne baze, pri čemer model in njegova logika ostajata nespremenjena. Zbirke podatkov ter ORM pristop bomo podrobneje opisali malo kasneje.

3.4.2 Pogled

Plast pogleda [13] je področje, kjer se Magento sistem oddalji od ostalih MVC aplikacij. Pogled je pri Magento sistemu razdeljen na tri različne komponente:

- **Postavitve (angl. Layouts):** Postavitev definirajo XML datoteke, kjer določimo blokovno strukturo, predloge in tudi uporabljene CSS in JavaScript knjižnice.
- **Bloki (angl. Blocks):** Bloki so v Magento sistemu uporabljeni za razbremenitev dela krmilnika. Vsebujejo na primer delčke HTML kode, ki se ustvari na podlagi podatkov iz baze.
- **Predloge (angl. Templates):** Predloge so *.phtml* datoteke, kjer lahko uporabimo tako HTML kot PHP kodo in je inicializirana preko blokov.

XML datoteke postavitve so tisto, kar v sistemu Magento prinese izjemno fleksibilnost, pri prikazu strani. Vsak modul ima svojo množico postavitvenih datotek, ki določajo, kaj se prikaže ob določenem zahtevku. Na sliki 3.4 je prikazano, kako Magento MVC arhitektura ustvari pogled.



Slika 3.4: Blokovni diagram Magento MVC arhitekture [13].

3.4.3 Krmilnik

V Magento MVC arhitekturi so krmilniki načrtovani kot redki krmilniki. To pomeni, da so načrtovani tako, da vsebujejo čim manj poslovne logike in so uporabljeni predvsem za zajem zahtevkov spletne aplikacije. Po zajemu so bloki odgovorni za izvedbo poslovne logike, pridobitev podatkov iz modelov ter pripravo podatkov za posredovanje pogledu [14].

3.5 ORM pristop in zbirke podatkov

Preden so PHP razvijalci prevzeli idejo MVC arhitekture, je bil dostop do podatkov izveden neposredno z uporabo SQL poizvedb in/ali s pomočjo abstraktne SQL plasti. Pisale so se SQL poizvedbe brez pozornosti na objekte, katere modelirajo.

Sodobne PHP aplikacije prevzemajo ORM pristop. ORM pristop je tehnika programiranja, ki omogoča pretvorbo podatkov med sicer nekompati-

bilnimi sistemi določanja tipov programskih konstruktov v objektno usmerjenih programskih jezikih. Ustvari "navidezno bazo objektov", ki jo lahko uporabljamo znotraj programskega jezika. Razvijalec ima opravka striktno z objekti, podatki pa se preko objektov samodejno zapišejo v določeno podatkovno bazo [15].

3.5.1 Anatomija modela v Magento sistemu

V prejšnjem podpoglavju smo že omenili, da Magento sistem uporablja modele za delo s podatki. Delijo se torej na dva tipa, preproste modele in EAV modele. Ne glede na ta tip pa se Magento model deli še na tri različne plasti:

- **Razred Model:** Tukaj se nahaja večina poslovne logike. Uporablja se za manipulacijo podatkov, vendar do njih direktno ne dostopa.
- **Razred Resource Model:** Resource Model priskrbi podatke našemu modelu in je zadolžen za CRUD (Create, Read, Update, Delete) operacije, torej kreiranje, branje, posodobitev in brisanje.
- **Razred Collection Model:** Zbirka modela je namenjena številnim individualnim instancam Magento modela.

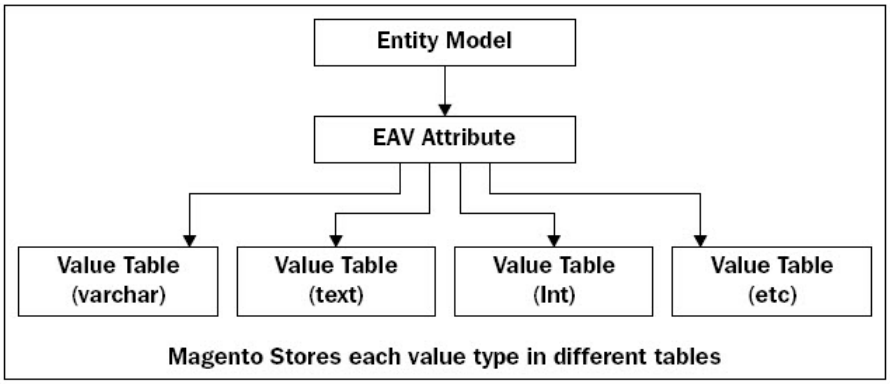
Magento modeli torej ne vsebujejo logike za komunikacijo s podatkovno bazo. V ta namen služi plast virov. Na ta način lahko Magento sistem vzpostavimo na različnih platformah in z uporabo različnih tipov podatkovnih baz. Čeprav je trenutno uradno podprta samo MySQL podatkovna baza, lahko razred z viri ustvarimo na podatkih katerekoli druge podatkovne baze, do katere je možno dostopati s programskim jezikom PHP.

Magento ORM uporablja eno najmočnejših funkcionalnosti jezika PHP in sicer metodo *__call()*. Splošne metode, uporabljene znotraj Magento modela, se uporabljajo za nastavitev (*set*), odstranitev (*unset*), preverjanje (*has*) in pridobitev (*get*) podatkov. Kadar želimo priklicati metodo, ki sicer ni definirana v trenutnem razredu, bo klic sledil vsem staršem razreda vse do klica funkcije *__call()* v razredu *Varien_Object*, ki kot argument prejme ime

metode ter njene argumente. Osnovnih funkcij, ki jih model potrebuje za obvladovanje podatkov baze, torej ni potrebno posebej definirati[16], kar bomo prikazali na primeru naše aplikacije.

3.5.2 EAV Model

Kratica EAV v EAV modelu označuje entiteto, atribut in vrednost (angl. Entity-Attribute-Value). Ta implementacija spada med bolj kompleksne v Magento sistemu, čeprav koncept ni narejen posebej za Magento sistem. Primer Magento EAV modela je prikazan na sliki 3.5.



Slika 3.5: Magento EAV model.

- **Entiteta**: Z entiteto lahko predstavimo objekte znotraj Magento sistema, kjer ima vsaka entiteta svoj unikatni ključ.
- **Atribut**: Atributi so pravzaprav lastnosti objekta. Lahko so definirani v več različnih tabelah.
- **Vrednost**: Vrednost posameznega atributa.

EAV model je še ena izmed lastnosti, zaradi katere je Magento sistem tako zmogljiv in fleksibilen. Entitetam oziroma objektom omogoča dodajanje atributov brez spremembe kode ali predlog [17].

Poglavje 4

Uporabljena programska orodja in tehnologije

Magento sistem uporablja širok nabor programskih tehnologij. Omenili smo že, da je zgrajen v PHP programskem jeziku, uporablja MySQL podatkovno bazo in razširja Zend Framework, ki ga v tem poglavju ne bomo ponovno opisovali. Med splošna orodja oziroma jezike za izdelavo spletnih strani, ki jih uporablja tudi Magento sistem, spadajo HTML, CSS in JavaScript. Za naše potrebe smo dodali še nekatere knjižnice, kot sta na primer jQuery in Bootstrap. Omenili bomo tudi označevalni jezik XML, ki v Magento sistemu služi predvsem za konfiguracije. Med razvojem smo uporabili programska orodja in aplikacije, kot so integrirano razvojno okolje NetBeans, XAMPP za lokalno vzpostavitev spletne aplikacije, SVN repozitorij za nadzor različic ter zelo uporabna orodja za razvijalce brskalnika Google Chrome.

V tem poglavju bomo na kratko opisali posamezno orodje ter navedli, kje in zakaj smo v naši aplikaciji to orodje uporabili. Nekaj dejanskih primerov je prikazanih v petem poglavju skupaj z opisom izdelane aplikacije.

4.1 PHP

PHP [18, 19] je skriptni jezik, ki se izvaja na strani strežnika. Napisan je v programskem jeziku C, izdelal pa ga je Rasmus Lerdorf leta 1994 in je bil na začetku namenjen predvsem razvoju spletnih aplikacij. Kratica PHP je pomenila osebno domača stran (angl. Personal Home Page), sedaj pa se uporablja izraz PHP: Hipertekstni predprocesor (angl. PHP: Hypertext Preprocessor). Uporablja se lahko kot vmesnik v ukazni vrstici (CLI - Command Line Interface), ustvariti pa je mogoče tudi samostojno aplikacijo z grafičnim vmesnikom. Glavne značilnosti programskega jezika PHP so naslednje:

- **Izvajanje na strežniku:** PHP koda se ne izvaja na lokalnih strežnikih ali spletnih brskalnikih, temveč na gostiteljskem strežniku.
- **Možnost delovanja na različnih platformah:** PHP skripta se lahko izvaja na vseh operacijskih sistemih in spletnih strežnikih (Unix, Windows ter druge 32 oziroma 64 bitne platforme za katere obstaja interpreter).
- **Kombinacija s HTML kodo:** PHP ukaze in izraze umestimo znotraj HTML kode. Jedro Magento sistema vsebuje (z izjemo blokov) zgolj PHP kodo, kombinacija s HTML kodo pa je vidna v *.phtml* predlogah.

4.2 MySQL

MySQL [20, 21] je sistem za upravljanje relacijskih podatkovnih baz (angl. RDBMS - Relational Database Management System). MySQL je odprtokodni sistem, deluje na različnih platformah in je zelo popularen kot izbira podatkovne baze pri razvoju spletnih aplikacij. Za dostop do podatkov uporablja SQL (Structured Query Language) poizvedbe. Tudi pri ogromnih količinah podatkov je sistem izjemno hiter, kar doseže na različne načine, na primer z indeksiranjem in tudi vzporednim procesiranjem, seveda pa morajo

biti podatki zato tudi ustrezno strukturirani. Dostop je možen več uporabnikom hkrati.

4.3 CSS

CSS (angl. Cascading Style Sheets) [22, 23] je stilski jezik za določanje izgleda in oblike dokumentov, napisanih v označevalnem jeziku. Sem spadajo dokumenti kot so na primer HTML, XML, SVG (vektorska grafika) in XUL (angl. XML User Interface Language). Daleč najbolj pa se uporablja pri razvoju spletnih strani ter namiznih in mobilnih aplikacij.

Pred uveljavitvijo CSS za določanje stilov dokumentov, se je pri mnogo spletnih straneh za postavitev in oblikovanje uporabljala tabelarična struktura in takšen primer je tudi star sistem evidence dela zaposlenih, ki smo ga omenili v uvodu. CSS omogoča ločitev informacij za predstavitev stila od HTML kode (npr. postavitev, barva, pisava), ki se nahajajo v *.css* datoteki. Za element ali skupino elementov lahko določimo obsežno število pravil, ki določajo njihov izgled. Pravila lahko določimo tudi glede na razred ali identifikacijsko oznako elementov.

V zadnjih letih je CSS močno napredoval, še posebej s pojavom CSS3. Sedaj lahko z novo verzijo CSS ustvarimo dinamično in interaktivno stran, kar je bilo nekaj let nazaj možno samo s pomočjo programiranja v JavaScriptu. Dodamo lahko animirane prehode, transformacije, dogodke ob kliku ali prehodu elementa z miškinim kazalcem, z uporabo le nekaj vrstic CSS kode. Tudi vse bolj popularen odzivni dizajn, ki je namenjen prikazu na mobilnih napravah, lahko dosežemo zgolj s CSS.

4.3.1 Bootstrap

Bootstrap [24] je odprtokodna in brezplačna zbirka orodij za poenostavljeno izdelavo privlačnih in dinamičnih spletnih strani ter aplikacij. Vsebuje mnogo predlog za tipografijo, gumbe, tabele, navigacijo, in drugih komponent, ki služijo za prikaz in interaktivnost HTML vsebine za uporabnike. Sestavlja

jo množica CSS pravil, JavaScript ter jQuery kode in tudi zbirka pisav.

Bootstrap je ena izmed knjižnic, ki smo jih dodali v reduciran Magento sistem, tako da izgled temelji na tej zbirki orodij. Uporabljena je zgolj za osnovo, saj smo veliko CSS pravil in tudi JavaScript kode naknadno prilagodili našim potrebam.

4.4 JavaScript

JavaScript [25, 26] je interpretativni, objektno usmerjen programski jezik, ki se največ uporablja pri razvoju interaktivnih spletnih aplikacij. Izvaja se lokalno na strani uporabnika, interpreter pa je vgrajen v večino sodobnih brskalnikov. JavaScript temelji na prototipih, kar pomeni, da za ponovno uporabo obstoječih objektov ustvari klone, ki služijo kot prototipi. Uporablja tako imenovane prvo-razredne funkcije, ki jih lahko pošljemo kot argument funkcije, vrnemo kot vrednost funkcije, in shranimo v spremenljivko ali v podatkovno strukturo. Pomembno je omeniti tudi, da kljub poimenovanju nima bistvene zveze s programskim jezikom Java in je pravzaprav nastal iz jezika C.

Magento sistem ima že v osnovi precej obsežno zbirko JavaScript knjižnic. Služijo na primer za mnogo HTML gradnikov tako v zaledju kot tudi ospredju Magento sistema, na primer za implementacijo prevodov, za urejanje produktov in še mnogo več. Za nas še najbolj pomembna JavaScrpt knjižnica je *prototype* knjižnica, ki smo jo, tako kot Magento sistem, uporabili za AJAX poizvedbe. Veliko primerov uporabe JavaScripta v Magento sistemu smo nadomestili z Bootstrap zbirko orodij, ki deluje skupaj z jQuery.

4.4.1 jQuery in prototype knjižnici

Na kratko opišimo posamezno knjižnico:

- **jQuery:** Je najbolj popularna JavaScript knjižnica zgrajena z namenom lažjega skriptiranja HTML-ja. Služi lažjemu dostopu do DOM

(angl. Document Object Model) elementov, izdelavi animacij, obravnavi dogodkov in AJAX zahtevkom.

- **prototype:** Vsebuje aplikacijske programske vmesnike (API) za delo z AJAX zahtevki in izboru DOM elementov.

V naslednjem podpoglavju bomo opisali AJAX, ki je v naši aplikaciji kar pogosto uporabljen. Magento sistem v ta namen uporablja *prototype* knjižnico in mi nismo izjema. Zaradi uporabe obeh smo morali rešiti manjši konflikt, saj uporabljata enak znak za izbor DOM elementov in sicer dolar ($) znak. Za jQuery smo kot izbirni znak uporabili niz jQuery.

4.4.2 AJAX

AJAX [27] je kratica za asinhroni JavaScript in XML (angl. Asynchronous JavaScript and XML). Uporablja se za asinhrono pošiljanje in pridobitev podatkov iz spletnega strežnika s pošiljanjem HTTP in HTTPS zahtev preko JavaScript objekta XMLHttpRequest. Izraz asinhrono pomeni, da deluje v ozadju in med izvajanjem ne vpliva na izgled ali vedenje spletne strani za oči uporabnika. Ime je delno zavajajoče, saj podatke lahko obravnavamo tudi v drugih oblikah in ne le v XML (golo besedilo, JSON, HTML), prav tako pa je izvedba lahko sinhrona, a se ta zelo redko uporablja.

4.5 XML

Do sedaj smo že večkrat omenili XML [28] (angl. Extensible Markup Language), ki je pomemben sestavni del Magento tem in modulov. Uporablja se za definicijo modulov, konfiguracijo modulov, nastavitve modula v zaledju konfiguracije in postavitvi elementov za prikaz v brskalniku.

XML je označevalni jezik, ki določa skupek pravil za kodiranje dokumenta v formatu, ki je berljiv tako človeku kot tudi stroju. Načrtovan je z namenom, da poenostavi, posploši in poveča uporabnost pri prenosu po internetu.

Velike kose informacij združi v dobro strukturiran in organiziran dokument, ki pravzaprav lahko predstavlja celovito podatkovno strukturo.

4.6 Ostala orodja in pripomočki

Za zaključek tega poglavja si oglejmo še orodja, ki smo jih uporabili med izdelavo in so bistveno pripomogla k učinkovitosti razvoja.

- **NetBeans IDE:** NetBeans je orodje za razvoj programske opreme oziroma integrirano razvojno okolje (angl. Integrated Development Environment). Najbolj primerno je za razvoj Java aplikacij, vendar ponuja podporo tudi drugim jezikom. Primerno označi in zazna morebitne napake v sintaksi za vse uporabljene programske tehnologije razen MySQL. Omogoča tudi razhroščevanje PHP kode.

- **Google Chrome developer tools:** Razvijalska orodja so integrirana v večino sodobnih brskalnikov. Nam je bil najbolj všeč brskalnik Chrome. Je nepogrešljivo orodje za razvoj spletne aplikacije. Omogoča intuitiven pregled HTML elementov ter pripadajoče CSS kode, prav tako pa omogoča urejanje in razhroščevanje JavaScript kode. Vse to lahko delamo med samim izvajanjem. Za posebej koristno orodje se je izkazalo na primer pri razvoju odzivnega dizajna, saj lahko simuliramo prikaz spletne strani na mobilnih napravah.

- **XAMPP:** Brezplačno večplatformno odprtokodno orodje za vzpostavitev spletnega strežnika. Sestavlja ga Apache HTTP strežnik, MySQL podatkovna baza ter urejevalnik *phpMyAdmin*, interpreter za programska jezika PHP in Perl. XAMPP smo uporabili, da smo aplikacijo lahko razvijali na lokalnem strežniku,

- **TortoiseSVN nadzor različic:** TortoiseSVN [29] je klient za *Apache Subversion*, implementiran kot dodatek za Windows lupino (angl. shell). *Subversion* je brezplačen sistem za nadzor različic. Upravlja

datoteke in direktorije ter vse njihove spremembe skozi čas. Nadzor različic je pomemben pri razvoju kakršnekoli aplikacije, predvsem kadar pri razvoju sodeluje več oseb. Za razvoj naše aplikacije je bil uporabljen, ker smo jo izdelovali na dveh napravah hkrati. Vsaj dnevne spremembe smo naložili na oddaljen strežnik in jih na kratko dokumentirali.

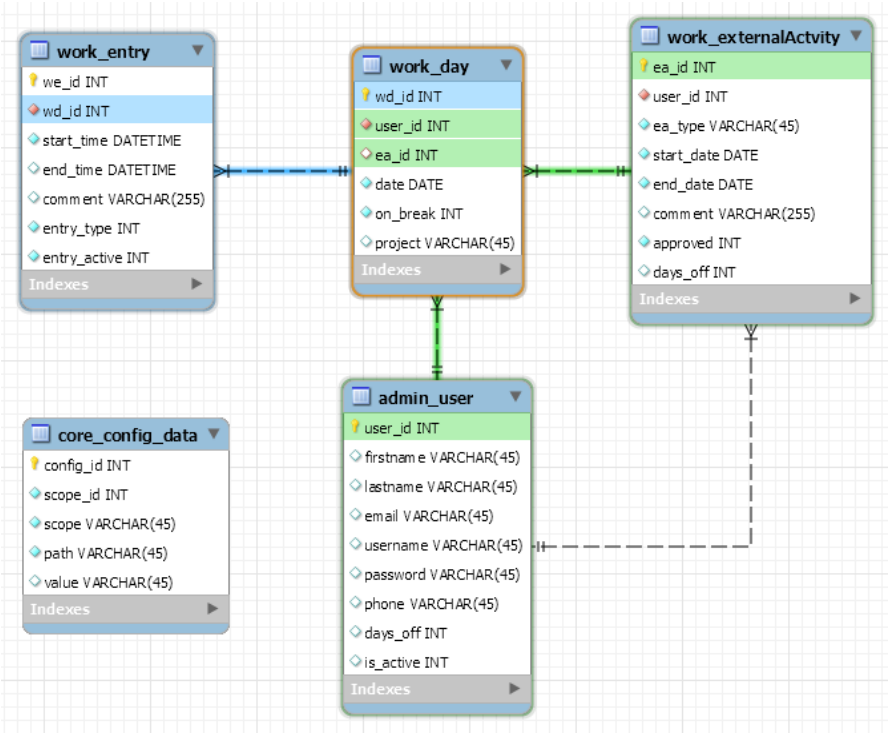
Poglavje 5

Postopek predelave Magento sistema v sistem za evidenco zaposlenih

Prvi korak pri predelavi Magento sistema je reduciranje sistema, kjer je cilj odstraniti vse sledi spletne trgovine. Pri tem smo morali paziti, da sistemu pustimo dovolj funkcionalnosti, da je razvoj modulov mogoč na enak način, kot je to mogoče v originalni verziji Magento sistema.

V tem poglavju bomo, na primeru našega glavnega modula, opisali postopek razvoja Magento modulov in posamezne gradnike poskusili čim bolj povezati z vsebino tretjega poglavja. Precej manj podrobno bomo opisali še ostale funkcionalnosti oziroma module našega sistema in prikazali nekaj zanimivejših rešitev. Za konec poglavja nam ostane še pregled izdelave dizajna.

Že pred reduciranjem sistema smo sestavili približno idejno zasnovo sistema za evidenco dela. Na sliki 5.1 je prikazan entitetno relacijski diagram našega sistema, ki ne vsebuje večine entitet reducirane verzije Magento sistema. Izjemi sta le entiteta uporabnikov zaledja (tabela *admin_user*), ki je direktno povezana z našim sistemom ter entiteta administrativne konfiguracije (tabela *core_config_data*), kjer je shranjena administrativna konfiguracija naših modulov in preostalih modulov Magento sistema.



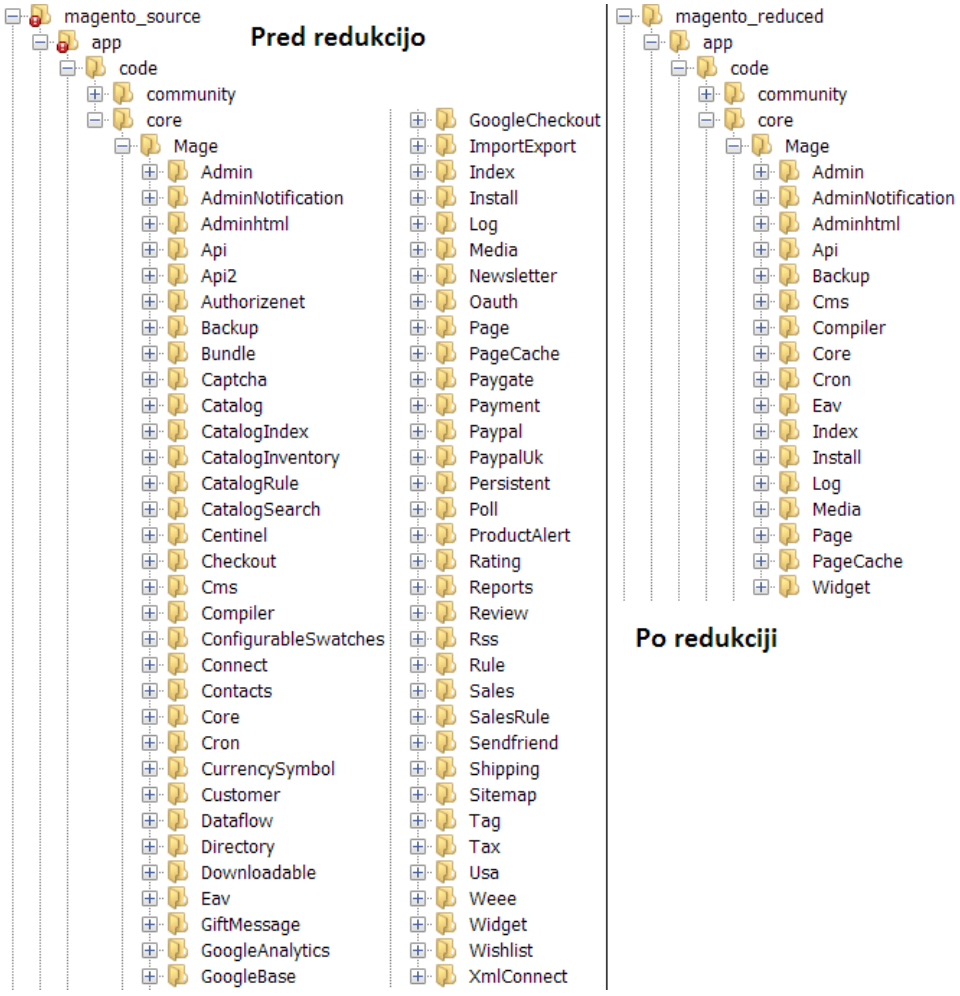
Slika 5.1: Entitetno relacijski diagram sistema za evidenco dela zaposlenih.

5.1 Reduciranje Magento sistema

Do sedaj smo že večkrat omenili modularnost in fleksibilnost Magento sistema. Kako dobro načrtovana in fleksibilna je modularna struktura Magento sistema, smo občutili že v prvem koraku predelave sistema. Reduciranje Magento sistema se je ravno zaradi teh lastnosti izkazalo za bolj enostavno, kot smo prvotno predvidevali. Večji del postopka reduciranja je bilo zgolj brisanje posameznih modulov.

Magento sistem, pri vzpostavitvi na strežnik, požene namestitveni postopek, ki v že obstoječi prazni podatkovni bazi generira vse potrebne tabele za posamezen modul. Izvedejo se vse procedure za kreiranje tabel v *sql* direktorijih modulov. Zahtevane podatke, torej ime baze ter podatke za dostop do MySQL strežnika, vnesemo ob namestitvenem postopku, ki se nato shranijo v novo ustvarjeno konfiguracijsko datoteko *app/etc/local.xml*.

Redukcije smo se lotili po namestitvi. Jedro kode vseh modulov se nahaja v direktoriju *app/code/core*. Za module, ki jih nismo potrebovali, smo izbrisali kar celoten direktorij. Vsebina *core* direktorija pred in po redukciji je vidna na sliki 5.2 levo oziroma desno. Kateri moduli so aktivni v Magento sistemu, določajo XML konfiguracijske datoteke v direktoriju *app/etc*. Vsebujejo informacije tudi o morebitni odvisnosti med moduli, na kar smo bili pri brisanju še posebej pozorni. Nekateri moduli imajo svoje XML datoteke, glavni pa so definirani v datoteki *Mage_All.xml*.



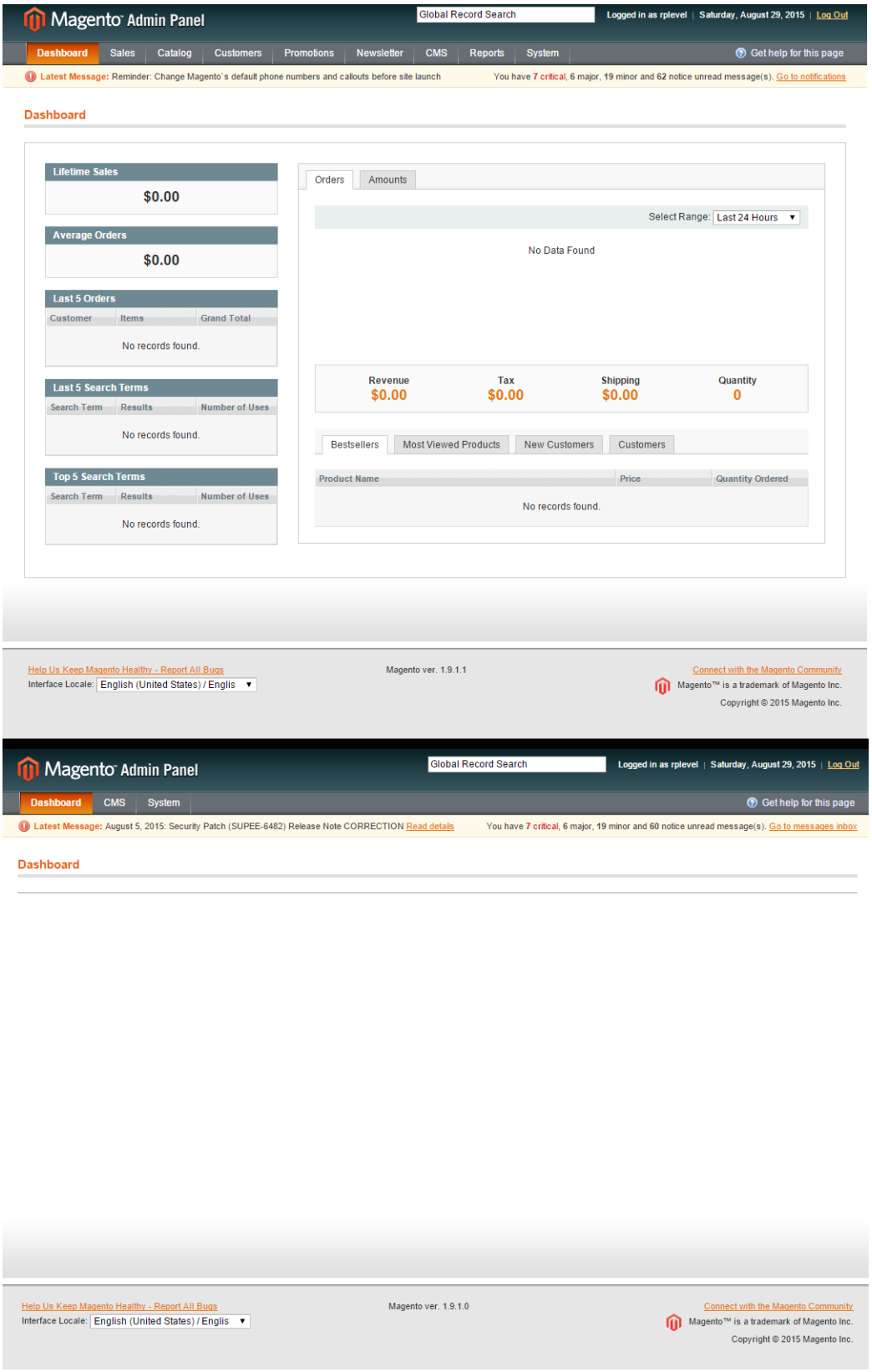
Slika 5.2: Primerjava jedra Magento sistema pred (levo) in po (desno) redukciji.

Izbrisati je bilo potrebno tudi datoteke za prikaz modulov v direktoriju *app/design*. Ta direktorij se deli na tri dele in tudi tukaj so datoteke razdeljene glede na modul:

- **adminhtml**: Vsebuje predloge in postavitvene datoteke za zaledje Magento sistema.
- **frontend**: Vsebuje predloge in postavitvene datoteke za ospredje sistema. Vsebuje tri različne teme, pri čemer smo obdržali samo temo *rwd*, ki je vklopljena ob prvi namestitvi sistema Magento.
- **install**: Vsebuje predloge in postavitvene datoteke za namestitveni proces. Tukaj smo spremenili le toliko, da smo odstranili podatke, ki jih zahteva namestitev, namenjene spletni trgovini.

Po končanem brisanju smo se lotili testiranja. Zagnali smo zaledje reduciranega Magento sistema in počakali, da sistem javi napako, nato pa smo sledili izvoru in izbrisali ali popravili delček kode, ki je uporabljal katerega izmed izbrisanih modulov. Pri tem nam je bil v veliko pomoč tudi modul *Mage_Log* za beleženje sistemskih napak in izjem. Nahajajo se v direktoriju *var/log/* v datoteki *exception.log* ter *system.log*.

Nazadnje smo z nizi imen izbrisanih modulov preiskovali vsebine datotek v celotnem direktoriju Magento sistema in iskali morebitne preostale reference na izbrisane module. Ko je bilo to končano, smo izbrisali ustvarjeno bazo ter *local.xml* konfiguracijsko datoteko, ter ponovno zagnali namestitveni postopek. Po uspešni namestitvi je bilo reduciranje Magento sistema zaključeno. Slika 5.3 prikazuje zaledje Magento sistema pred (zgoraj) in po (spodaj) redukciji.



Slika 5.3: Zaledje Magento sistema pred in po redukciji.

5.2 Razvoj modulov v Magento sistemu

V tem podpoglavju bomo prikazali in razložili postopek razvoja modulov v Magento sistemu na primeru razvoja glavnega modula. Glavni modul v našem sistemu za evidenco dela zaposlenih je vmesnik za prihod na delo, odhod na odmor ter zaključek dela. Ob prihodu zaposleni izbere projekt, na katerem trenutno dela, na koncu pa napiše še komentar o opravljenem delu. Na takšen način deluje že obstoječ sistem za evidenco dela zaposlenih pri podjetju, za katerega smo razvijali sistem za evidenco dela. Naš modul se imenuje *Work* in uporablja dva modela in sicer *Entry* ter *Day*. Modela upravljata s tabelama *work_day* in *work_entry* v podatkovni bazi (slika 5.1).

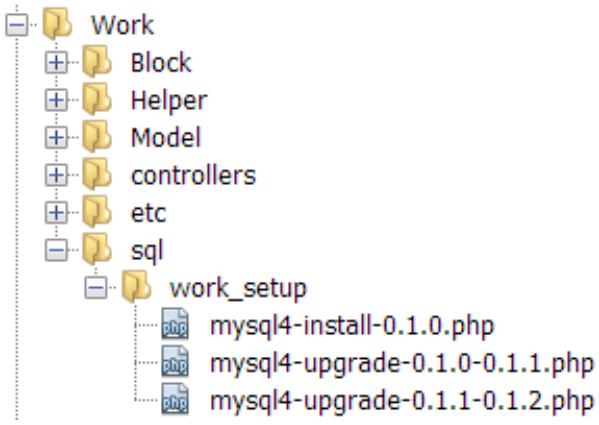
5.2.1 Vzpostavitev kode in splošna konfiguracija modula

Pri vzpostavitvi kode moramo najprej določiti kodno območje našega modula, določiti imenski prostor in ime modula. Naš modul se nahaja v kodnem območju *local* in enako velja tudi za vse ostale module, ki služijo sistemu za evidenco dela. Tudi imenski prostor je enoten za vse module in se imenuje *Worksystem*. Potrebujemo še ime modula. Naš glavni modul se imenuje *Work*. Vse te informacije navedemo v konfiguracijski datoteki *Worksystem_Work.xml* znotraj direktorija *app/etc/modules*. Vsebina datoteke je prikazana na sliki 5.4.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <config>
3      <modules>
4          <Worksystem_Work>
5              <active>true</active>
6              <codePool>local</codePool>
7              <version>0.1.2</version>
8          </Worksystem_Work>
9      </modules>
10 </config>
```

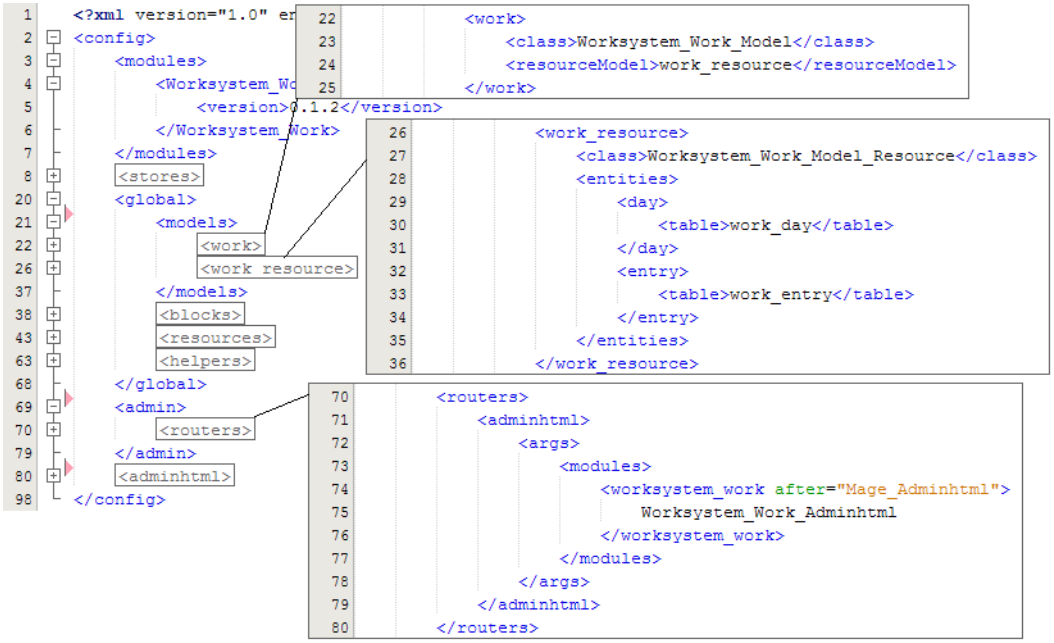
Slika 5.4: Konfiguracijska datoteka za aktivacijo *Work* modula.

Na sliki 5.4 lahko vidimo tudi oznako *version*. To je informacija o verziji modula, ki je pomembna predvsem za nadgradnjo podatkovne baze. Na začetku tega poglavja smo omenili namestitveni postopek, kjer se generira podatkovna baza. V bazo se zapiše tudi verzija modula. Če želimo spremeniti podatkovno bazo brez izgube podatkov, moramo pripraviti skripto za nadgradnjo, ki se nanaša na verzijo modula. Princip poimenovanja skript za namestitev in nadgradnjo je viden na sliki 5.5. V našem primeru smo nadgrajevali samo strukturo tabel (npr. *mysql4-upgrade-0.1.1-0.1.2*), lahko pa bi nadgradili tudi podatke (npr. *data-upgrade-0.1.1-0.1.2*).



Slika 5.5: Namestitvene in nadgradne skripte Magento modula

Najprej smo govorili o konfiguracijski datoteki, ki jo potrebuje Magento sistem. Sedaj si oglejmo še konfiguracijo, ki je pomembna specifično za naš modul. Definirana je v datoteki *Work/etc/config.xml*.



Slika 5.6: Konfiguracijska datoteka za lastnosti znotraj modula.

Na sliki 5.6 je prikazana strnjena konfiguracijska datoteka za naš glavni *Work* modul. Na kratko si poglejmo pomen nekaterih sklopov:

- **<modules>:** Glavne informacije o modulu.
- **<stores>:** Znotraj te značke povemo, katero temo uporablja modul. Več bo pojasnjeno v zadnjem poglavju.
- **<models>, <resources>, <blocks>, <helpers>:** Znotraj teh značk definiramo kje znotraj direktorija modula najdemo skupine modelov, virov, blokov in pomožnih razredov.
- **<work_resource>:** V tej znački definiramo entitete modula, ki se nanašajo na tabele v podatkovni bazi.
- **<routers>:** Znotraj te značke definiramo URL za usmeritev do modula. Pomembno za krmilnik in postavitvene datoteke.

- **<adminhtml>:** Informacije o zaledju modula (datoteka postavitve, prevodi ...).

5.2.2 Prikaz modula uporabniku

Kodo za prikaz našega modula uporabnikom, lahko razdelimo na tri dele in sicer na bloke, postavitvene (angl. layout) datoteke in predloge (angl. templates). Bloki so definirani znotraj glavnega direktorija modula, saj so namenjeni prikazu elementov glede na podatke iz podatkovne baze. Predloge in postavitvene datoteke se nahajajo v direktoriju *design*. Na sliki 5.7 je prikazana postavitvena datoteka *design/adminhtml/default/worksystem/layout /worksystem_work.xml*. Razložimo najprej pot do datoteke. *Adminhtml* pomeni, da gre za zaledje Magento sistema. *Default* je trgovina, *worksystem* pa naš pogled trgovine. Več o trgovinah in pogledih trgovin bo razloženo v šestem poglavju.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <layout version="0.1.0">
3      <default>
4          <reference name="head">
5              <action method="addJs"><script>worksystem/work_entry.js</script></action>
6              <action method="addJs"><script>worksystem/work_status.js</script></action>
7              <action method="addCss"><name>clock/clock.css</name></action>
8              <action method="addJs"><script>worksystem/bootstrap-datetimepicker/bootstrap-datetime
9              <action method="addCss"><name>bootstrap/css/datetimepicker/bootstrap-datetimepicker.m
10         </reference>
11         <reference name="right">
12             <block type="work/entry" name="work_entry" template="worksystem/work/entry.phtml" />
13         </reference>
14     </default>
15     <adminhtml_status_index>
16         <reference name="content">
17             <block type="work/status" name="work_status" template="worksystem/work/status.phtml"
18         </reference>
19     </adminhtml_status_index>
20 </layout>
```

Slika 5.7: Postavitvena datoteka glavnega modula.

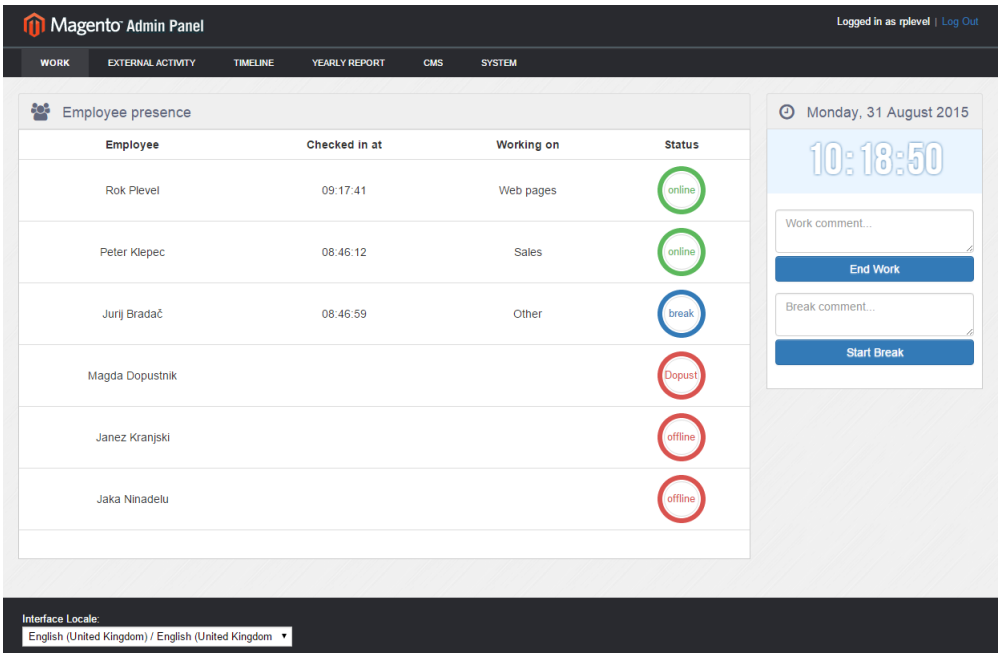
Postavitvena datoteka vsebuje referenco na mesta na strani, ki so definirana v še eni postavitveni datoteki *main.xml*. Sklicujemo se na tri dele na glavni postavitvi:

- **default**: Vsebina znotraj tega sklopa je dodana na vse strani našega sistema.
    1. **head:** Sklop je namenjen CSS in JavaScript datotekam, ki jih uporabljajo različni moduli. V *main.xml* je že nekaj obstoječih referenc.
    2. **right:** Na desni predel strani postavimo vmesnik za prijavo, odjavo in premor dela. Naš modul inicializira blok *Worksystem_Work_Block_Entry*, ter nanj veže še predlogo.
- **adminhtml_status_index**: Vsebina tega sklopa je dodana samo kadar URL ustreza imenu. URL se določi v *routers* sklopu konfiguracijske datoteke modula in nadaljuje glede na blok ali krmilnik.
    1. **content:** V tem območju je uporabljen blok *Worksystem_Work_Block_Status*, ki prav tako uporablja predlogo.

Razred, ki definira blok, in dodeljena *.phtml* predloga sta neposredno povezani. Vzemimo za primer blok *Entry*. Predloga *entry.phtml* med upodabljanjem kliče funkcijo, definirano v razredu bloka. Blok iz podatkovne baze priskrbi podatke o aktivnem delovnem dnevu, če ta obstaja, in pošlje podatke predlogi, ki nato ustvari vmesnik za delovne vnose. Na sliki 5.8 je stran, ki jo uporabniku prikaže naš glavni modul.

5.2.3 Krmilnik in AJAX zahtevki

Akcije, ki jih sproži uporabnik preko vmesnika, so obravnavane v krmilnikih. Magento sistem do krmilnikov dostopa preko URL-ja. Del URL-ja določa sklop *routers*, ki se nahaja znotraj konfiguracijske datoteke modula.



Slika 5.8: Stran glavnega modula, ki se v zaledju sistema prikaže uporabniku.

Poglejmo si primer URL-ja *swsys/index.php/admin/entry/start*:

1. *swsys*: Direktorij v katerem se nahaja celota našega sistema. Spredaj manjka še ime domene.
2. *index.php*: Vsaka stran se inicializira preko datoteke *index.php*.
3. *admin*: Definirano v *routers* sklopu konfiguracije modula.
4. *entry*: Razred krmilnika je definiran v *EntryController.php*.
5. *start*: Ime akcije, kar pomeni klic funkcije *startAction* znotraj krmilnika, ki začne delovni dan.
6. Na koncu URL-ja Magento sistem definira še unikaten ključ za vsako akcijo.

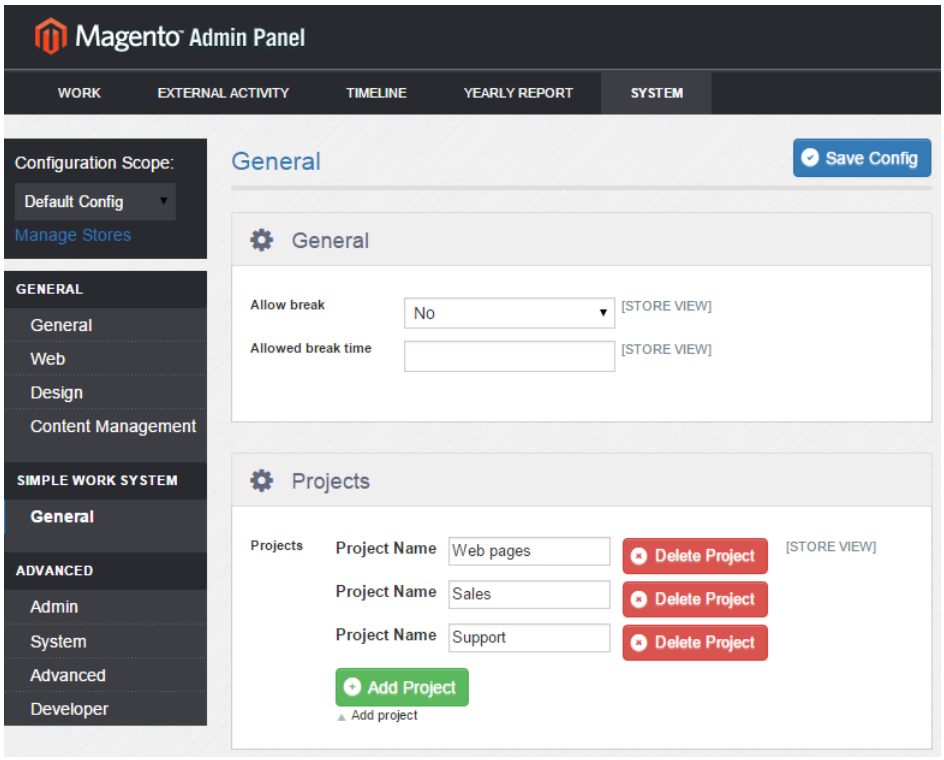
Primer krmilnika je prikazan na sliki 5.9. Najprej je definirana funkcija *preDispatch*, ki definira spremenljivke, ki jih potrebuje večina funkcij krmilnika. Sledi že omenjena funkcija *startAction*. Funkcija preko modela zapiše ustrezne podatke v bazo in na podlagi zapisanih podatkov se ustvari nov blok. Akcijo krmilnika kličemo s pomočjo AJAX zahtevka, ki prejme vsebino bloka v HTML obliki, ter jo na novo prikaže. S tem se izognemo postopku osveževanja strani, kar bi pomenilo ponovno generiranje čisto vseh blokov na strani. V primeru napake med izvajanjem nam sistem prikaže ustrezno sporočilo.

```php
<?php
class Worksystem_Work_Adminhtml_EntryController extends Mage_Adminhtml_Controller_Action
{
    protected $_userId;
    protected $_date;
    protected $_datetime;

    public function preDispatch()
    {
        parent::preDispatch();

        $dateModel = Mage::getModel('core/date');

        $this->_date = $dateModel->date('Y-m-d');
        $this->_datetime = $dateModel->date('Y-m-d H:i:s');
        $this->_userId = Mage::getSingleton('admin/session')->getUser()->getUserId();

        return $this;
    }

    public function startAction()
    {
        try
        {...18 lines } catch (Exception $e) {...4 lines }

        $this->loadLayout();
        $block .= $this->getLayout()->createBlock('Worksystem_Work_Block_Entry','right',
                                                   array('template' => 'worksystem/work/entry.phtml'))->toHtml();
        $this->getResponse()->setBody($block);
    }
```

Slika 5.9: Primer krmilnika za obravnavo dogodkov, ki jih sproži uporabnik.

5.2.4 Konfiguracija modula na administrativni strani

Za možnosti konfiguracije modula na administrativni strani sistema, uporabljajo moduli datoteko *etc/system.xml*. Magento sistem ima obsežno zbirko HTML elementov, ki jih uporabimo v konfiguraciji in definiramo v sistemski datoteki (npr. input, textarea, select). Primer konfiguracije vidimo na sliki 5.10.



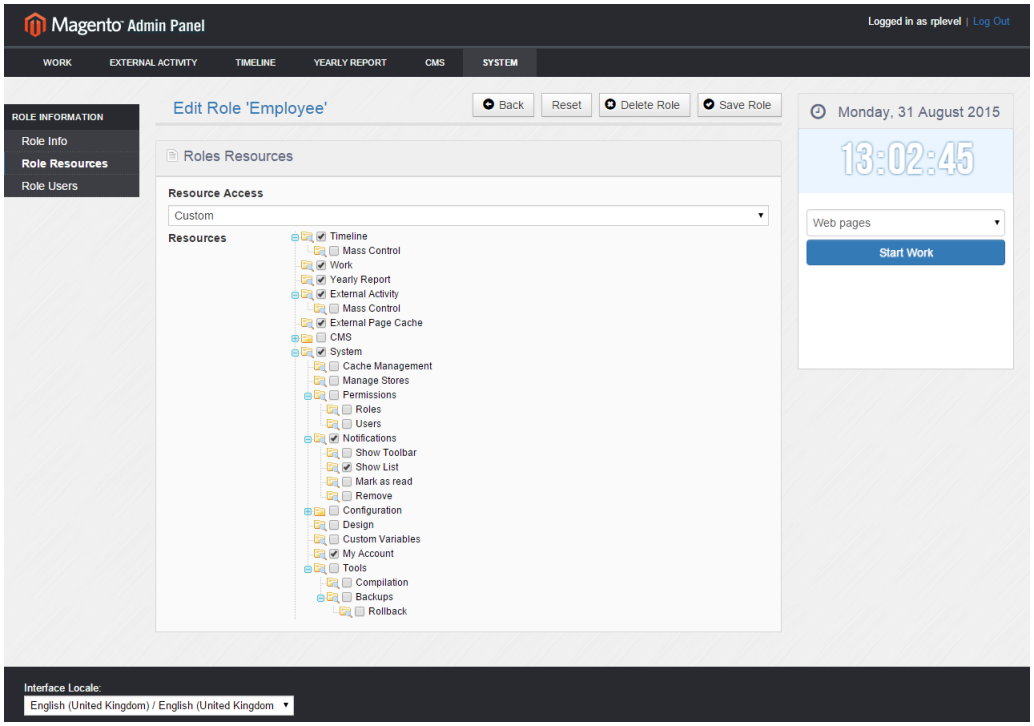
Slika 5.10: Konfiguracija modulov na administrativni strani.

V *system.xml* datoteki *Worksystem_Work* modula smo definirali zavihek (<*tab*>) "simple work system", sklop (<*section*>) "splošno" ter skupini (<*group*>) polj "splošno" in "projekti". Za polja (<*field*>) definiramo tipe (npr. select, text), lahko pa tudi validacijo pri polju za dovoljen čas premora. Malo drugačna je konfiguracija projektov, ki jo uporablja Magento model zaledja, imenovan *serialized_array*. Za to polje smo definirali tudi blok, kjer definiramo polja znotraj tabele in gumbe za dodajanje in brisanje vnosa v tabeli. Do konfiguracij za, na primer tabelo projektov, dostopamo z ukazom:

```
Mage::getStoreConfig('work_general/projects_group/add')
```

5.2.5 Uporabniki administrativne strani in njihova dovoljenja

Eden izmed glavnih razlogov, da smo se, pri izdelavi sistema za evidenco dela zaposlenih, lotili zaledja Magento sistema, je seznam kontrole dostopov (angl. Access Control Lists). Jedro Magento sistema vsebuje modul *Mage_Acl*. Dovoljenja temeljijo na uporabnikih, ki jim dodelimo različne vloge. Ustvarili smo vlogo za zaposlene z omejenim dostopom do funkcionalnosti, in vlogo za administratorje s polnim dostopom do funkcionalnosti. Dodelitev virov vlogi zaposlenih je prikazana na sliki 5.11.



Slika 5.11: Dodelitev virov vlogi zaposlenih v podjetju.

Tukaj pride v poštev še zadnja izmed konfiguracijskih datotek modula in sicer *etc/adminhtml.xml*. Nekaterim modulom smo v tej datoteki nastavili sklop kontrol "Mass Control". Na objektu uporabnikove seje preverimo, ali ima uporabnik dovolj pravic z naslednjo kodo:

```
$session->isAllowed('worksystem/timeline/control');
```

5.3 Razširitev jedra Magento sistema

Magento sistem ponuja možnost razširitve dela že obstoječega modula. Primer razširjenega modula v našem sistemu je modul zaledja *Adminhtml*, kjer želimo uporabniku dodati število dni dopusta in telefonsko številko. Izdelali smo modul *Worksystem_Coreextended*. V njegovi konfiguracijski datoteki (*config.xml*) določimo dele modula, ki jih razširjamo:

- **Razširitev krmilnika:** V konfiguracijski datoteki na novo definiramo sklop <routers>. Razred krmilnika deduje originalni krmilnik, popravili pa smo funkcijo za shranjevanje, da obravnava tudi dopust in telefonsko številko.

- **Razširitev blokov:** V sklopu <blocks> podamo ime našega modula ter še en sklop <rewrite>, kjer povemo, kateri blok v našem modulu razširja blok prvotnega modula.

- **Razširitev podatkovne baze:** Modulu dodamo navadno SQL skripto. V našem primeru razširimo tabelo *admin_user*.

Na podoben način kot blok, lahko v Magento sistemu razširimo tudi modele in pomožne razrede.

5.4 Pregled funkcionalnosti sistema za evidenco dela

V prejšnjem poglavju smo med postopkom izdelave Magento modula spoznali glavnega izmed naših modulov. Modul je namenjen dnevnemu vnosu in pregledu statusa zaposlenih v realnem času. Priskrbi podatke, katere kasneje, v daljšem časovnem obdobju, uporabljajo drugi moduli. Vsi ostali moduli so

odvisni od tega modula, kar je tudi navedeno v konfiguracijski datoteki za aktivacijo modula v Magento sistemu.

Implementacije smo se lotili tako, da vsak modul predstavlja svoj zavihek menija in je tudi v svojem direktoriju znotraj imenskega prostora *Worksystem*. Preostali moduli so še:

- *Worksystem_Timeline*: Modul je namenjen prikazu mesečne časovnice, kjer lahko zaposleni vidijo dnevne zapise svojih aktivnosti. Prikaže dnevno število opravljenih ur, čas prihoda in odhoda, morebitne odmore in dnevne komentarje dela. Modul je, z vidika administratorja, ki lahko časovnico tudi ureja, prikazan na sliki 5.12.

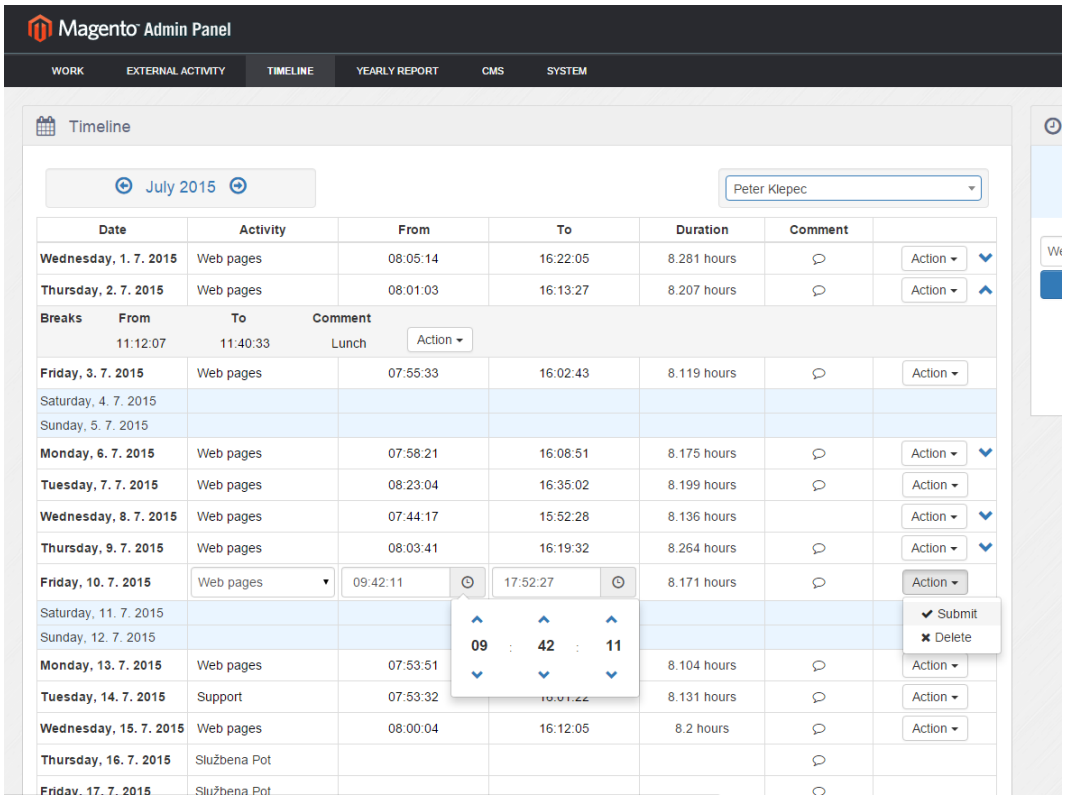
- *Worksystem_ExternalActivity*: Modul služi aktivnostim, ki spadajo izven običajnega delovnega dne. Tukaj zaposleni rezervirajo dopuste, javijo bolniško ter označijo morebitno službeno pot. Aktivnost se postavi na čakanje za odobritev. Slika 5.13 prikazuje ta modul z vidika vloge zaposlenega (zgoraj) in administratorja (spodaj).

- *Worksystem_Yearly*: Modul je namenjen prikazu letne časovnice, ki je strukturirana podobno kot mesečna, z mesečnim številom opravljenih ur, številom ur na odmoru, porabljenimi dnevi dopusta in bolniškimi dnevi.

- *Worksystem_Coreextended*: Glavna razširitev je omenjena v prejšnjem podpoglavju. Ostale razširitve, ki jih lahko najdemo v tem modulu, so namenjene še strani za prijavo in ponekod, zaradi HTML kode znotraj blokov, tudi za dosego želenega dizajna.

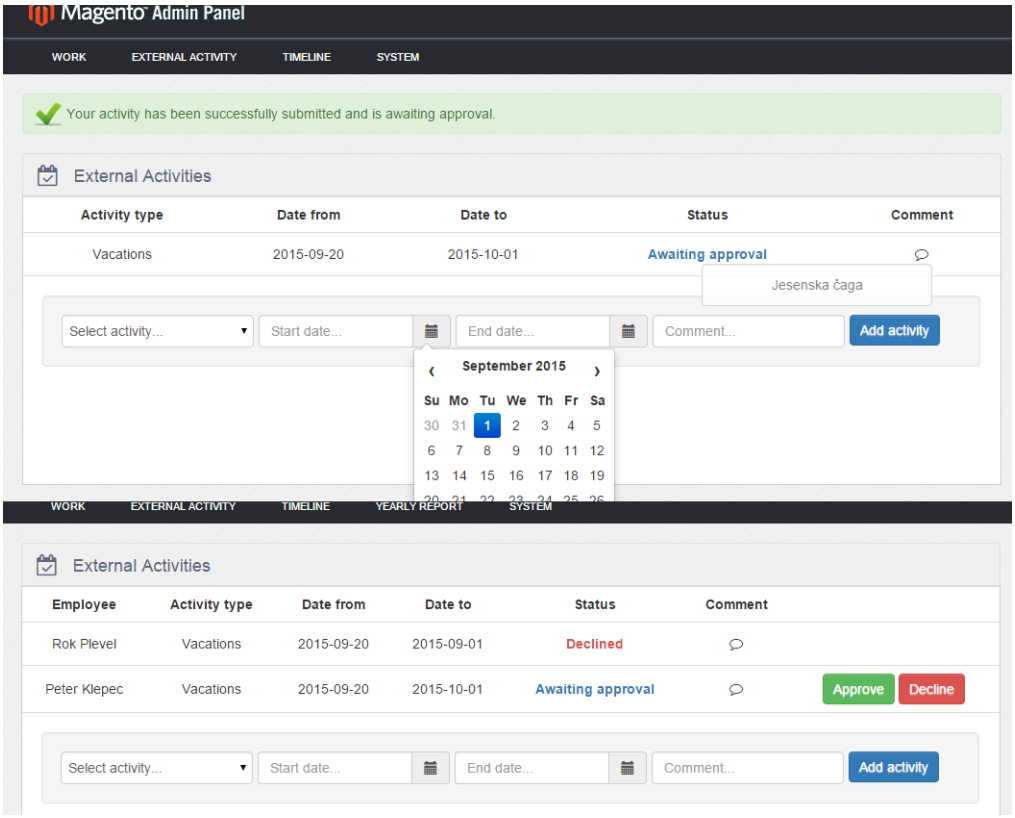
V tem poglavju smo že omenili administratorske vloge. Pri našem glavnem modulu administratorska vloga nima nikakršnega pomena, ima pa kar precejšen pomen v modulih za zunanje aktivnosti ter mesečno časovnico. Pri zunanji aktivnosti smo implementirali funkcionalnost za odobritev in zavrnitev aktivnosti. Takšno pravico ima lahko samo administratorska vloga.

Zaposleni vidijo samo svoje aktivnosti, administrator pa vidi čakalno listo aktivnosti vseh zaposlenih. Na podoben način deluje tudi mesečna časovnica. Mesečno časovnico ima administrator možnost urejati. Prikazan ima tudi vmesnik, ki omogoča izbor kateregakoli zaposlenega.



Slika 5.12: Mesečna časovnica z administratorskega vidika.

Preostanek funkcionalnosti večinoma implementira že Magento sistem. Sem spada celotna konfiguracija, vmesnik za prijavo v sistem za evidenco dela, uporabniške vloge in računi uporabnikov. V nogi Magento sistema je ponujena tudi možnost menjave jezika. Nize v naši aplikaciji prikažemo z uporabo pomožnega razreda Magento sistema, ki služi prevodom. V ta namen smo pripravili datoteke z vrednostmi ločenimi z vejicami (CSV - Comma Separated Values). Predstavljajo vrednost niza in slovenski prevod. V zadnjem poglavju bomo govorili še o zavihku menija CMS, ki je viden administrator-



Slika 5.13: Zunanje aktivnosti z vidika vloge zaposlenega in administratorja.

jem strani.

5.5 Dizajn

Zadnji korak pri razvoju sistema za evidenco dela je še dizajn. Zgledovali smo se po temi Realm, omenjeni v drugem poglavju, ki jo ponujajo razvijalci Bluethemes. Velik del te teme je predstavljen z Bootstrap knjižnico, ki je tudi v našem sistemu ključnega pomena za dizajn. Knjižica je bila sicer deležna kar nekaj sprememb, vendar je osnovni princip prikaza še vedno precej podoben originalu. Vnosna polja, sistemska sporočila, okna, v katera je postavljena vsebina in ikone, ki so pravzaprav tip pisave, so oblikovani s pomočjo Bootstrap knjižnice.

Za izgled sistema v veliki meri poskrbi CSS. Pri Magento sistemu se CSS datoteke nahajajo v *skin* direktoriju, kjer se nahaja tudi Bootstrap. Znotraj tega direktorija smo ustvarili še *worksystem* paket, na katerega se sklicujemo preko glavne konfiguracijske datoteke modulov, torej *config.xml*. Da posamezni elementi prevzamejo CSS pravila Bootstrap knjižnice, je potrebno DOM elementom dodati ustrezne razrede. Magento sistem ima pripravljenih mnogo CSS datotek za dizajn, ki se nahajajo v *skin/default* direktoriju. Pravila teh datotek lahko povozijo tista, ki so nastavljena v Bootstrap knjižnici, zato smo jih morali kar nekaj preurediti. Preurejene CSS datoteke, in tudi nekatere slike, z istim imenom vstavimo v naš *skin/worksystem* direktorij in Magento sistem bo namesto privzetih vzel te.

Rezultat našega dizajna je viden na več zaslonskih slikah, ki smo jih prikazali zgoraj.

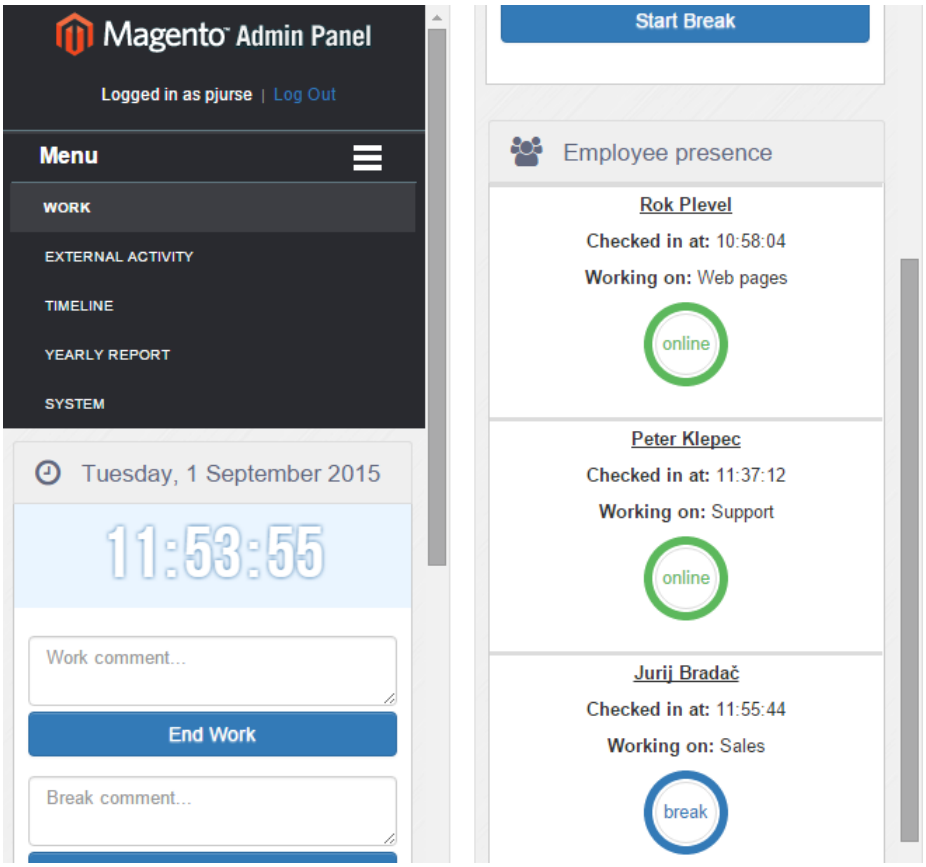
5.5.1 Odzivni spletni dizajn

Odzivni spletni dizajn (angl. Responsive Web Design) je pomemben dejavnik sodobnih spletnih strani ter aplikacij in omogoča dobro uporabniško izkušnjo na napravah z različnimi velikostmi zaslonov. Za sistem za evidenco dela sicer ni bistvenega pomena, prav pa pride zaposlenim, ki občasno delajo na terenu in morajo do aplikacije dostopati preko naprav z manjšimi zasloni, pogosto so to mobilni telefoni.

Odzivni dizajn je pristop spletnega oblikovanja, ki uporabnikom prinese optimalno izkušnjo pregleda in interaktivnosti na vseh velikostih zaslona. Še posebej pride v poštev pri mobilnih napravah, kjer želimo uporabniku omogočiti uporabo aplikacije brez pomikanja po širini ter bližanja in oddaljevanja po zaslonu.

Takšen dizajn dosežemo s tekočo mrežo elementov, katerih dimenzije definiramo z odstotki in ne piksli ali točkami. Veliko je že doseženo z Bootstrap knjižnico, vendar smo morali določene stvari implementirati sami. V pomoč so nam tudi CSS *media* poizvedbe, ki aktivirajo CSS pravila glede na širino zaslona. Na sliki 5.14 je prikazana statusna stran našega sistema na ekranu

širokem približno 300 pikslov. Bistvena sprememba je glavni meni, ki je zložen v en sam gumb, ki se ob kliku razkrije. Desni vmesnik se postavi na vrh strani. Celice v tabeli statusov se razširijo čez celo stran in so prikazane vertikalno.



Slika 5.14: Odzivni dizajn strani na ožjem zaslonu.

Poglavje 6

Sklepne ugotovitve in nadaljnji razvoj

V diplomski nalogi sta bila zastavljena dva glavna cilja. Prvi cilj je bila predelava Magento sistema za upravljanje vsebin za spletne trgovine v bolj splošen sistem za, z vidika razvijalcev, poenostavljeno izdelavo spletnih aplikacij, ki temelji na principu Magento MVC arhitekture. Predelava sistema nam je tukaj dobro uspela, kar se je izkazalo med razvojem sistema za evidenco zaposlenih. Malokrat se je zgodilo, da bi se nam sistem zaradi kakšnih nepredvidenih razlogov sesul, v redkih primerih, ko pa se je to zgodilo, pa smo brez večjih težav napako popravili. Razvoj je potekal tako, kot smo si na začetku zastavili in nikjer nismo našli situacije, kjer bi nas Magento sistem s čemerkoli omejeval.

Drugi cilj je bila izdelava sistema za evidenco dela zaposlenih. Tudi to je bilo uresničeno. Vendar pa bi bilo potrebno, za uporabo razvitega sistema tudi v praksi, še temeljito preverjanje za obstoj morebitnih napak, prav tako pa bi bilo ponekod potrebno implementirati sredstva, ki poskrbijo, da so podatki zapisani v bazi, konsistentni. Sistem vključuje vse funkcionalnosti, ki so bile prisotne pri starem sistem in še kar nekaj dodatnih funkcionalnosti. Nekaj izmed teh funkcionalnosti smo tudi predstavili v predhodnem poglavju. Funkcionalnost, ki nam je še ni uspelo implementirati, in je pravzaprav ne

implementira niti obstoječi sistem, je izvoz podatkov o zaposlenih, ki jih je nato možno uvoziti v računovodski sistem, ki ga uporablja podjetje. Ustvariti bi bilo potrebno ustrezen vmesnik, s katerim bi določili katere podatke se izvaža, ter pripraviti model, ki pridobi podatke iz baze, ter s pomočjo orodji jezika PHP ustvari XML datoteko, primerno za uvoz v računovodski sistem. Izdelava modula, ki bo to omogočal, je predvidena v nadaljnjem razvoju.

6.1 Nadaljnja uporaba obstoječih funkcionalnosti Magento sistema

Že na začetku smo predvidevali, katere funkcionalnosti Magento sistema bi lahko uporabili za namen novega sistema za evidenco dela zaposlenih. Nekatere ideje za dodatne funkcionalnosti sistema za evidenco dela pa so se pojavile tudi sproti. Želeli smo reducirati obstoječi sistem v bolj splošen sistem, ki bi, poleg izdelave sistema za evidenco dela, omogočal tudi še nadaljnje dopolnitve in predelave, kot so:

- **Administrativna sporočila:** Eden izmed modulov, ki smo ga v sistemu obdržali je modul *AdminNotifications*. Služi marketingu in obvestilih o posodobitvah Magento sistema. Obvestila se pojavijo na administrativni strani, z uporabo zgoščenega povzetka spletne strani (RSS - Rich Site Summary). Predelali bi ga tako, da bi administratorji sestavili obvestilo za vse zaposlene.

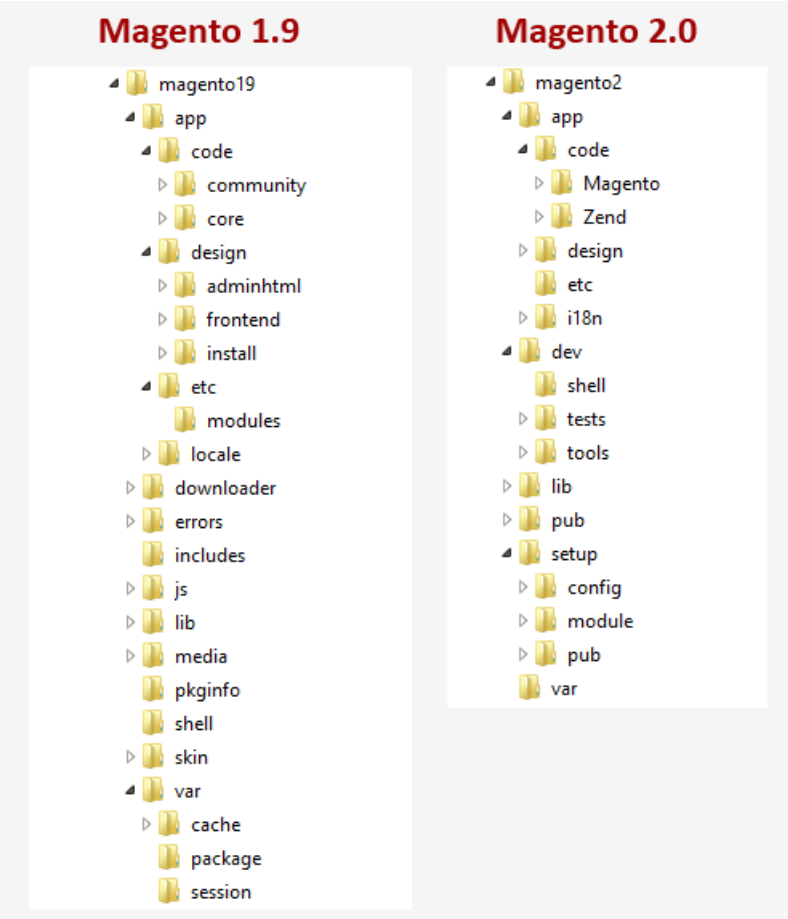
- **Trgovine in pogledi trgovine**: Magento sistem je lahko implementiran na treh nivojih: različna domena (*website*), različne trgovine z drugimi izdelki (*stores*), različni pogledi trgovine (*store views*). Trgovine ali pa morda poglede trgovine bi preuredili v oddelek podjetja. Konfiguracija modulov je že prilagojena temu, tako da bi že lahko imeli različne projekte in aktivnosti. Težava bi bila omejiti posamezno trgovino ali pogled na administratorsko vlogo.

- **CMS za ospredje strani:** Trenutno je na ospredju samo slepa stran, ki preusmeri uporabnika na administrativno stran. V predelanem sistemu še vedno lahko razvijemo modul za ospredje po naših željah, vendar je tukaj cilj urediti bolj enostavno izdelavo ospredja. To je v Magento sistemu možno na zavihku CMS, kjer lahko generiramo stran s svojo postavitvijo (XML format) in tudi posamezen blok, ki ga nato na stran vključimo. Ko smo odstranili modul *Mage_Catalog* smo na ospredju izgubili meni. Katalog izdelkov je bil namreč zavihek menija. Zaradi obsežnega nabora funkcionalnosti kataloga, bi bilo morda lažje zavihke menija razviti po svoje.

6.1.1 Posodobitve in Magento 2

Problem, ki nam še ostaja pri predelanem sistemu, so posodobitve. Trenutno so posodobitve sistema možne samo ročno. Posodobitev poteka tako, da vzamemo zadnjo verzijo izvorne kode Magento sistema ter pregledamo za spremembe v reduciranem jedru naše kode. Morda posodobitve preko uradne strani niti ne bi bile potrebne in bi jih v primeru težav implementirali sami. Posodobitev preko uradne verzije Magento sistema bodo še posebej težavne po izidu verzije Magento 2.

Beta verzija za Magento 2 je bila na Git-u izdana 15. julija, kar je bilo za nas prepozno, da bi se predelave lotili tudi v tej verziji, ki vključuje kar nekaj sprememb. Princip MVC arhitekture je sicer enak, vendar je struktura direktorijev, prikazana na sliki 6.1, zelo drugačna. Spremenjeno je tudi delovanje postavitvenih datotek, CSS poprocesiranje in še bi lahko naštevali [30].



Slika 6.1: Primerjave datotečne strukture Magento 2 in 1.9.x.

Literatura

[1] (2015) Statistika uporabe mobilnih naprav, dostopno na:
http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[2] (2015) Zakonom o evidencah na področcju dela in socialne varnosti, dostopno na:
https://www.uradni-list.si/1/content?id=72976

[3] (2015) Magento sistem za upravljanje vsebin za elektronsko poslovanje, dostopno na:
http://magento.com/

[4] (2015) Obstoječ sistem AllHours, dostopno na:
http://allhours.com/

[5] (2015) Obstoječ sistem Realm, dostopno na:
http://www.bluthemes.com/themes/realm/

[6] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003

[7] (2015) Zend Framework, dostopno na:
http://framework.zend.com/about/

[8] (2015) SOLID: Principi objektno usmerjenega programiranja in načrtovanja, dostopno na:
https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)

[9] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003, str. 28

[10] (2015) Magento MVC arhitektura temeljena na konfiguraciji, dostopno na:
http://devdocs.magento.com/guides/m1x/magefordev/mage-for-dev-1.html

[11] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003, str. 42

[12] (2015) Slika: Elementi in potek delovanja Magetno MVC arhitekture, dostopno na:
http://devdocs.magento.com/common/images/m1x/Magento_MVC.pdf

[13] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003, str. 43-45

[14] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003, str. 46

[15] (2015) ORM pristop programiranja, dostopno na:
https://en.wikipedia.org/wiki/Object-relational_mapping

[16] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003, str. 59-62

[17] Allan MacGregor, *Magento PHP Developers Guide*, Packt Publishing, 2003, str. 65

[18] (2015) Opis tehnologije PHP, dostopno na:
https://en.wikipedia.org/wiki/PHP

[19] Knowledge flow, *Learning PHP, MySQL, JavaScript, and CSS*, Knowledge flow, 2015, str. 7-10

[20] (2015) MySQL, dostopno na:
http://dev.mysql.com/doc/refman/5.7/en/introduction.html

[21] Knowledge flow, *Learning PHP, MySQL, JavaScript, and CSS*, Knowledge flow, 2015, str. 27

[22] (2015) CSS, dostopno na:
https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[23] Robin Nixon, *Learning PHP, MySQL, JavaScript, CSS & HTML5*, O'Reilly Media, 2014, str. 9

[24] (2015) Bootstrap, dostopno na:
https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[25] Javascript, dostopno na:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[26] Robin Nixon, *Learning PHP, MySQL, JavaScript, CSS & HTML5*, O'Reilly Media, 2014, str. 8

[27] AJAX zahtevki, dostopno na:
https://en.wikipedia.org/wiki/Ajax_(programming)

[28] (2015) XML, dostopno na:
https://en.wikipedia.org/wiki/XML

[29] (2015) TortoiseSVN, dostopno na:
http://tortoisesvn.net/about.html

[30] (2015) magento2, dostopno na:
http://www.ubertheme.com/magento-news/11-exciting-features-magento-2/